UNIVERSIDADE DO MINDELO
DEPARTAMENTO DE ENGENHARIA E RECURSOS DO MAR

CURSO DE LICENCIATURA em INFORMÁTICA DE GESTÃO

PROJETO DE LICENCIATURA
ANO LETIVO 2017/2018 – 4º ANO

Autor: Kelvin Jonhson Nascimento da Cruz, N.º 2848

Orientador: Dr. SAMUEL LIMA

Mindelo, 2018

Kelvin Johnson Nascimento da Cruz

SERVIÇOS SOS – APLICAÇÃO MÓVEL PARA PEDIDO DE SOCORRO

Trabalho de Conclusão de Curso apresentado à **Universidade do Mindelo** como parte dos requisitos para obtenção do grau de Licenciatura em Informática de Gestão.

Orientador:
Dr. Samuel Lima

Mindelo, 2018

DEDICATÓRIA

Dedico este trabalho a todos que acreditaram em mim nessa jornada académica, a minha família pelo apoio incondicional e aos meus amigos, colegas e professores.

AGRADECIMENTO

Primeiramente agradeço a Deus pela força e coragem para a realização deste trabalho.

Agradeço aos meus familiares, principalmente meus Pais que me apoiaram nos momentos bons e menos bons nessa caminhada.

Ao meu Orientador Samuel Lima um obrigado pela paciência e disponibilidade que transmitiu durante esse tempo, pelas sugestões impostas nos momentos de decisões importantes.

A todos os meus professores, amigos e colegas de turma com que convivi nesses anos de curso.

A todos um especial abraço e muito obrigado pelo apoio.

“Eu faço a dificuldade a minha motivação. A volta por cima, vem na continuação.”

(Charlie Brown Jr)

ÍNDICE

ÍNDICE DE FIGURA

RESUMO

O Mundo hoje está rendido a novas tecnologias principalmente as aplicações móveis, apresentado um crescimento consideravelmente no mercado e a sociedade de uma forma ou de outra tem que acompanhar esses avanços tecnologicos.

A sociedade está a se tornar cada dia mais violenta e as novas tecnologias pode ajudar a estancar essa violência e outros acontecimentos desde que seja usado de forma adequada logo surgiu a ideia de criar algo que ajudasse a minimizar esses acontecimentos.

Com a criação desta aplicação pode dar suporte a esses acontecimentos, a ideia é ter uma aplicação onde o utilizador possa usar de uma forma rápida e eficaz e com um tempo de espera reduzido de forma a evitar constrangimentos, e a entidade que é transmitida as informações ou dados precisos do utilizador.

Com armazenamento das informações gravadas na nuvem, através de uma página Web consegue aceder-se aos dados necessários.

A ferramenta como JSON (*Objects JavaScript*), cuja finalidade é apresentar estrutura de dados de uma maneira compreensível para que possam utilizar esse sistema.

Paralelamente surge Firebase onde fica armazenado todas as informações necessárias.

Para tratamento dessas informações que são enviadas é preciso um Front-end onde a entidade responsável vai fazer o tratamento dos dados e dar sequência as ocorrências enviadas pelo utilizador.

Esta aplicação é desenvolvida para plataforma Android chamado “Serviços SOS”, foi desenvolvido para dar resposta as ocorrência solicitadas pelo utilizador, Firebase e responsável pelo todo o armazenamento do sistema.

Palavras-chaves: Firebase, Android, Ocorrência dos utilizadores, Frond-end, JSON.

ABSTRACT

The world today is surrendered to new technologies mainly the mobile applications, presented a considerable growth in the market and the society of one form or another has to follow these technological advances.

Society is becoming more and more violent and new technologies can help stave off this violence and other events of which it is used properly. Soon the idea came up to create something that would help to minimize these events.

With the creation of this application can support these events, the idea is to have an application where the user can use quickly and effectively and with a waiting time reduced in order to avoid constraints, and the entity that is transmitted information or or accurate data of the user.

By storing information recorded in the cloud, a web page can access the data you need.

The tool like JSON (Objects JavaScript), whose purpose is to present data structure in an understandable way so that they can use this system.

At the same time Firebase is where all the necessary information is stored.

In order to handle this information, a Frond-end is needed where the responsible entity will process the data and sequence the occurrences sent by the user.

This application is developed for Android platform called "Services SOS", was developed to respond to the occurrence requested by the user, Firebase and responsible for the entire system storage.

Keywords: Firebase, Android, User Instance, Frond-end, JSON.

LISTA DE SIGLAS E ABREVIATURAS

JSON	JavaScript Object Notation
GPS	Global Positioning System
API	Application Programming Interface
JVM	Java Virtual Machine
SGBD	Sistema de Gerenciamento de Banco de Dados
SMS	Short Message Service
VM	Virtual Machine
NFC	Near Field Communication
APP	Applicative
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
APK	Android Package Kit
IDE	Integrated Development Environment
BPMN	Business Process Modeling Notation
UML	Unified Modeling Language
GPL	General Public License
FCM	Firebase Cloud Massage
URL	Uniform Resource Location
MVVM	Model-View-View-Model
XML	Extensible Markup Language
WEB	World Wide Web
SDK	Software Development Kit

CAPÍTULO I

1. Introdução

Na criação de novas tecnologias proporcionou um papel importante na evolução da sociedade, permitindo maior prática, rapidez e eficiência. Os benefícios alcançados em diversos sectores tais como Medicina, Transportes, Meios de Comunicação, indústria, Comércio, e entretenimento.

Com uma aplicação móvel é possível ter um rápido acesso a informação, troca de mensagens, etc. Os *Smartphones* tornaram-se um recurso indispensável na vida quotidiana, com tudo isso identificar oportunidades necessárias para criação e acompanhamento dos avanços tecnologicos.

No mercado as tecnologias tem sido imprescindível no desenvolvimento económico. O investimento nos serviços de telecomunicação tem-se aumentado significativamente no mundo e Cabo Verde está a acompanhar essas novas tendências.

Atualmente as instituições confrontam-se com o ambiente de rápido e constante mudança ou seja deparam com um entorno global, cada vez mais competitivo e de avanços tecnologicos constantes.

Cabo Verde tem sido o palco para apresentação de novas tecnologias e vai continuando a ser sempre, sabendo que cada dia novas atualizações estão disponíveis para uma melhor adaptação.

O principal objetivo desta aplicação e trazer mais avanços para as instituições que podem ser aplicadas no seu meio, mas cada dia que passa o mercado está a ficar mais competitivo, por conseguinte aumenta o grau de exigência perante a aplicação que é desenvolvida de forma saber dar resposta a serviços que lhe é solicitado.

2. Objetivo

2.1 Objetivo Geral

É desenvolver um protótipo onde envolve serviços tais como Polícia, Bombeiro, Hospitais, Hotelaria e Serviços de Aluguer de Viaturas, onde um determinado utilizador pode solicitar um determinado serviço de emergência de forma a contactar as entidades competentes necessárias sem ser pelo modo convencional neste caso uma chamada telefónica.

2.2 Objetivo Específico

Solicitar serviços para uma melhor resposta a um tempo reduzido e mais precisão com o objetivo de minimizar os riscos de acidentes ainda piores.

- Esta aplicação é virada extremamente pela parte de emergência acima referida anteriormente;
- Na aplicação existe uma vertente onde o utilizador pode fazer uma escolha de qual serviço pretende solicitar;
- Informa ao utilizador quais ocorrência que pretende escolher;
- O utilizador pode receber mensagens do responsável pelo controlo dos pedidos de socorro;
- Exibir ocorrências em tempo real as ocorrência assim solicitadas e recolha de informações do utilizador;
- A entidade pode recolher informações da sua localização atual.

3. Motivação

Hoje em dia na sociedade deparamos com situações improváveis onde sentimos a necessidade de ultrapassá-las e com a utilização dos dispositivos móveis podemos minimizar algumas dessas situações.

A sociedade Cabo-verdiana está ciente do que tem vindo a acontecer nesses últimos tempos, tendo em conta que essa aplicação pode trazer uma mais-valia para a nossa sociedade onde o utilizador tem possibilidade de solicitar serviços de emergência em tempo real que pode diminuir alguns constrangimentos.

Criar uma Aplicação onde podemos solicitar serviços de socorro é uma mais-valia para o utilizador onde tais necessidades são enormes.

Experiência vivida numa empresa turística levou-me a realização desse projeto tais como uma dificuldade enorme em dar resposta aos pedidos solicitados a um espaço curto de tempo.

4. Metodologia

Para inicializar este projeto foram realizadas pesquisas sobre o conteúdo em si que serão tratados tais como plataforma Android, Firebase, vertente de localização mais concretamente a parte de GPS.

Pesquisas realizadas em torno desse projeto ou seja se já existe algo em concreto no mercado Cabo-verdiano, de modo que possa ser desenvolvido sem nenhum impasse na sua elaboração.

Foi desenvolvido um protótipo para dispositivo de sistema Android onde integra algumas funcionalidades que podem ajudar um cidadão no seu dia-dia.

Implementar um sistema de armazenamento de dados inclui diversas funcionalidades mais concretamente a plataforma Firebase onde podemos armazenar dados, esse sistema de armazenamento interliga dispositivos Android com a parte Web.

Através dessa aplicação as entidades tornar-se-ão mais eficientes e eficazes na localização dos utilizadores, tentando alcançar os objetivos traçados. Sabendo que as aplicações móveis estão a invadir o mercado Cabo-verdiano. Com essa aplicação o utilizador tem possibilidade de pedir ajuda em qualquer lugar desde que esteja ligado a internet.

5. Estrutura do trabalho

Este trabalho se encontra estruturada em seis capítulos, primeiramente uma breve Introdução, os objetivos, motivação, metodologia.

No segundo capítulo será abordado parte teóricas tais como, estrutura geral Google Android, plataforma Android, Arquitectura Android, componentes da aplicação Android, ciclo de vida de uma aplicação, versões da plataforma Android, coordenadas geográficas.

No terceiro capítulo será abordado as ferramentas e tecnologia no desenvolvimento do protótipo.

No quarto capítulo será abordado as análise do sistema a desenvolver.

No quinto capítulo será abordado a parte de funcionamento de protótipo, onde vai ser demonstrado os passos para o manuseamento das funcionalidades que se encontra no protótipo.

No sexto capítulo será abordado as conclusões do respetivo projeto.

E por fim o sétimo capítulo será abordado sobre os Trabalhos Futuros.

CAPÍTULO II

6. Enquadramento Teórico

Como já sabemos a sociedade vem ganhando grandes contornos em todos os sentidos, onde temos que acompanhar essas mudanças.

A segurança das pessoas tem que estar a um nível onde consegue dar resposta às necessidades, e por isso as entidades responsáveis tem que saber como lidar com determinadas situações e o que fazer de modo que tenha um controle melhor.

Com Serviços SOS conseguimos ter um controlo dos pedidos assim solicitados onde temos as informações necessárias para chegar a esse utilizador sem muita demora.

O mais importante é reduzir o tempo de espera, de modo geral, a segurança das pessoas é extremamente importante em caso de emergência.

Com esse protótipo pode-se minimizar alguns riscos tais como o tempo de resposta que pode ser reduzido, a localização pode ser mais ou menos precisa devido a determinadas circunstâncias, onde pode ser uma vantagem enorme para as entidades responsáveis.

E por fim com todas as informações recolhida pelas entidades disponível e posto em prática ou seja dar resposta aos pedidos assim solicitados pelos utilizadores e garantido a segurança dos demais.

7. Plataforma Android

7.1. Android

Segundo o Autor Luciano Alves da Silva (Apostila de Android, Programando Passo – Passo, 7º Edição), Android é o nome do sistema operativo baseado em *Linux* em que é utilizado em dispositivos móveis tais como *Smartphones*, *Tablet*.

É desenvolvido pelo *Open Handset* e várias empresas, entre elas a Google.

O funcionamento do Android é idêntico a outros sistemas operativos tais como *Windows*, *Mac OS*, *Ubuntu*, entre outros, a sua função é gerir todos os processos das aplicações e de *Hardware* de um computador para que funcionam em perfeitas condições.

A diferença é que o Android foi impulsionado pela Google para ser utilizado nos próprios dispositivos móveis e desta forma entra numa concorrência com outros

sistemas operativos dominantes tais como Symbian (dispositivos Nokia), iOS (dispositivos *Apple* como a *IPhone*) e *Blackberry OS*.

8. Estrutura Geral da plataforma Google Android

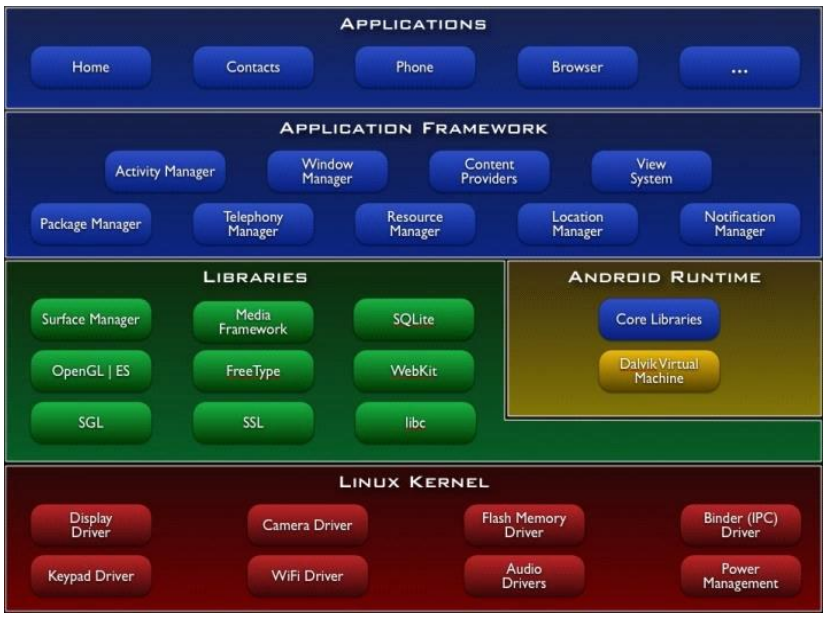
Segundo o Autor Luciano Alves da Silva (Apostila de Android, Programando Passo – Passo 7º Edição), Android SDK é uma ferramenta de desenvolvimento que disponibiliza um conjunto de *APIs* necessário para desenvolver aplicações para a plataforma Android utilizando a linguagem Java.

Recurso encontrados nessa plataforma:

- **Application Framework:** Permite a reutilização e substituição de componentes;
- **Dalvik Virtual Machine:** é uma máquina virtual Java (JVM) voltada para dispositivos móveis;
- **Browser integrado:** Baseado no *Website engine*;
- **Gráficos otimizados:** O Android é instituído, as bibliotecas 2D e 3D baseada na especificação *OpenGL ES 1.0*;
- **SQLite:** Sistema gerenciado de Banco de Dados (SGBD) já embutido no Android para guardar dados;
- **Suporte Multimídia:** A plataforma já oferece para áudio, vídeo e formatos de imagem (MPEG4, H.264, MP3, AAC, AMR, JPG, GIF);
- **Telefone GSM:** (Dependente de *Hardware*);
- **Bluetooth, EDGE, 3G e Wi-Fi:** (Dependente de *Hardware*);
- **Camera GPS**, compasso e acelerómetro: (Dependente de *Hardware*);
- **Rico ambiente de desenvolvimento**: Incluindo um emulador de dispositivos, ferramentas de depuração memória, performance e um *plugin* para o eclipse.

9. Arquitetura do Android

Figura 1: Arquitetura do Android



Fonte: Apostila de Android (Programando Passo – Passo 7º Edição de Luciano Alves da Silva)

9.1. Aplicações

Android nos fornece um conjunto de aplicações fundamentais:

- Um cliente de Email;
- Programa de SMS;
- Agenda;
- Mapas;
- Navegador;
- Contactos entre outros.

Todas as aplicações acima presentes no Android foram desenvolvidas na linguagem de programação Java.

9.2. Biblioteca

O Android nos fornece um conjunto de bibliotecas C/C++ utilizados por vários componentes do sistema:

- **System C library:** Consiste em uma implementação derivada da biblioteca C padrão baseado no sistema (*libc*) do BSD sintonizada para dispositivos que rodam no *Linux*;

- **Midia libraries:** Baseado no *Packet* vídeos *Open CORE*, são as bibliotecas que suportam os mais diversos formatos de áudio e vídeo incluindo também imagens;
- **Surface Manager:** Responsável pelo acesso ao subsistema de exibição bem como as múltiplas camadas de aplicações 2D e 3D;
- **SQL** o engine de gráficos 2D;
- **3D libraries** uma implementação baseada no *OpenGL ES 1.0 APIs*;

As bibliotecas utilizam aceleração 3D via *Hardware* (quando disponível) ou o *Software* que renderá ação 3D altamente otimizado incluindo no Android:

- **Free Type:** Biblioteca responsável pela renderá ação de fontes *bitmap* e vetor;
- **SQLite:** Consiste no sistema gestão de base de dado (SGBD) relacional disponível para todas as aplicações.

9.3. Android Runtime

Segundo o Autor Luciano Alves da Silva (Apostila de Android, Programando Passo – Passo 7º Edição), Android é constituído por um conjunto de bibliotecas que fornece a maioria das funcionalidades disponíveis nas principais bibliotecas da linguagem Java.

Toda a aplicação Android roda em seu próprio processo com sua própria instância de máquina virtual *Dalvik*. O *Dalvik* foi escrito de forma a executar varia VMs eficientemente. Ele executa Arquivos.dex, que é otimizado para consumo mínimo de memória.

A VM é baseada em registos e roda classes compiladas pela linguagem *Java* que foram transformadas em arquivos.dx, através da ferramenta “dx” incluindo SDK.

9.4. Linux Karnel

Segundo o Autor Luciano Alves da Silva (Apostila de Android, Programando Passo – Passo 7º Edição), o Android já projetado em cima da versão 2.6 do karnel do Linux para os serviços centrais do sistema, tais como segurança, gestão de memória, gestão de processos, etc. O karnel também atua como uma camada de abstração entre *Hardware* e o resto do *Software*.

10. Componentes da aplicação Android

Componentes da uma aplicação são blocos de construção básicos de um aplicativo Android. Estes componentes são de baixo acoplamento pela aplicação, organização de arquivos de Manifesto. AndroidManifest.xml descreve cada componente da aplicação e como eles se interagem.

10.1. Atividades

Atividades representam um ecrã única com uma interface de utilizadores, onde uma aplicação de email pode ter uma atividade que mostra uma de novos emails, outra atividade que escreve e temos uma outra atividade que para leitura. Essas atividades funcionam juntas.

Exemplo: Uma classe de atividade se uma subclasse de atividade.

public class MainActivity extends Activity {

}

10.2. Service

Service são componentes executados em segundo plano para realizar operações de execução longa ou para realizar trabalho para processos remotos, mas representam uma interface do utilizador, ele pode estar em segundo plano quando a fazer diferentes buscas.

Outros componentes tais como atividades podem iniciar o serviço e deixar ele a executar ou interagir com ele.

Exemplo: *public class MyService extends Service {*

}

10.3. Content Provider

Content Provider é um conjunto de partilha de dados de uma aplicação. É possível armazenar os dados no sistema de aplicativos em uma base de dados *SQLite* ou em qualquer lugar de armazenamento persistente onde a aplicação possa aceder, por meio de um *Content Provider* outras aplicações que podem consultar ou até modificar os dados, se assim permitir.

O sistema Android oferece ao *Content Provider* gerenciar os dados de contato do utilizador. Qualquer aplicação com permissões adquiridas pode consultar parte do

Content Provider (como *ContactContrac.Data*) para leitura e gravar informações sobre uma determinada pessoa.

Exemplo*:* *public class MyContentProvider extends ContentProvider{*

}

10.4. Broadcast Receiver

Broadcast Receiver são componentes que respondem a anúncios de transmissão por todo o sistema. Muitas dessas transmissões se originam em sistemas.

Um aspecto exclusivo de um projeto do sistema Android é que qualquer aplicativo pode iniciar um componente de outra aplicação.

Quando um sistema inicia um componente, ele inicia o processo da aplicação que já não se encontra em execução e instanciar as classes necessárias para os componentes.

O sistema executa cada aplicação em uns processos separados com permissões de arquivos que restringem o acesso a outras aplicações. A aplicação pode ativar diretamente um componente de outra aplicação. No entanto o sistema Android pode fazer isso, ativar um componente em outra aplicação é preciso uma mensagem do sistema que especifique o *intent* de iniciar um componente específico, e em seguida o componente ativado.

Exemplo: *public class MyReceiver extends Broadcast Receiver {*

}

11. Ciclo de Vida de uma Aplicação

11.1. Visão geral das classes

Uma *activity* é uma tarefa muito focada no que um utilizador pode fazer. Quase todas as atividades interagem com o utilizador, uma classe toma conta da criação de uma janela onde poderá colocar todos os componentes de UI (*User Interface*) com *setContentView*. Enquanto atividades são normalmente apresentadas para o utilizador como ecrãs *full-screen*, também podem ser apresentadas de outra maneira: como janelas flutuantes (através de um tema com *WindoISFloating* configurado) ou embutida dentro de outra atividade (usando *ActivityGroup*).

Existem dois métodos que quase todas as subclasses de *Activity* são implementadas:

- **onCreate(Bundle)** - é onde se inicia as atividades. É chamada de *setContentView (int)* com recursos de *Layou*t definido a sua interface, e usar

findViewById (int) para retornar os *widgets* naquela interface que interage sobre forma de programa;

- **onPause()** – O utilizador deverá nesse ponto ser efetivado (utiliza *ContentProvider* que guarda os seus dados).

Para que possa ser usada pelo *Context.startActivity*, todas as classes de actividades devem ter uma declaração <*activity*> no manifeste toda aplicação em *AndroidManifest.xml*.

A classe *Activity* é mais importante no ciclo de vida de uma aplicação e a forma como as atividades são colocadas e unidas, é parte fundamental para a plataforma de modelo de aplicação.

11.2. Ciclo de vida

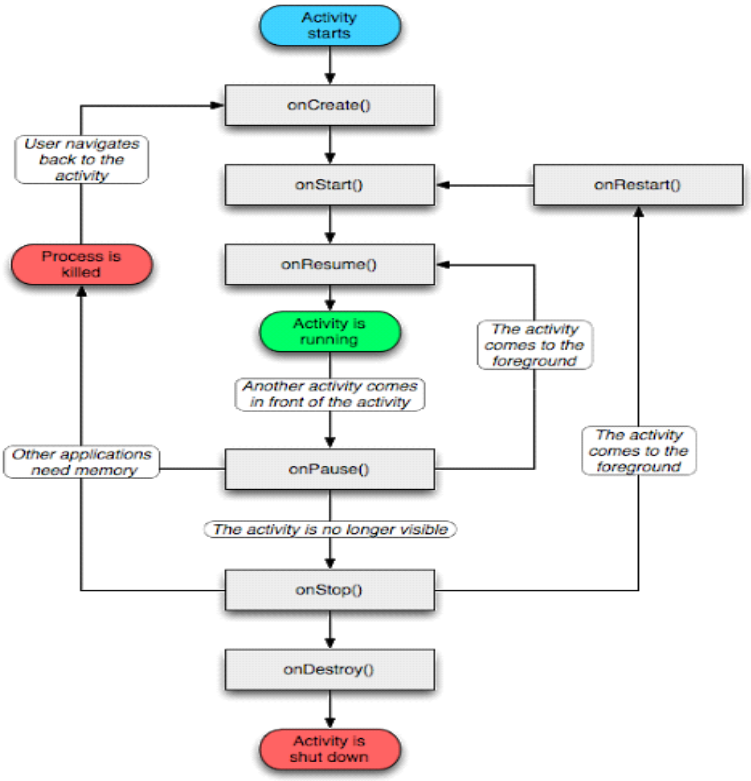
Atividades no sistema são gerenciadas como uma *activity stack*, ou pilha de atividades. Quando uma atividade é iniciada é colocada no topo da pilha e não vai ter amostra enquanto a atividade corrente não terminar.

Quatro estados de uma atividade:

- Se uma atividade está sendo executada e está sendo mostrado no ecrã (que é o topo da pilha), está em modo ative ou running;
- Quando uma atividade perde foco mas ainda está visível ou seja uma nova atividade está sendo mostrado no ecrã mas não ocupando completamente as atividades em janelas flutuantes;

  Por exemplo está em modo *paused*, está completamente viva, mantém todos os estados e informação de memória baixa;
- Se uma atividade é completamente obscurecido por outra atividade, está em modo *stopped*. Retém o seu estado e informações, com tudo não é mais visível pelo utilizador sua janela escondida pode ser que seja encerrada pelo sistema quando for necessário libertar memória;
- A atividade está em modo *paused* ou *stopped*, o sistema pode retirar atividade da memória simplesmente pedindo que seja finalizado ou simplesmente matando o seu processo. Quando é mostrado ao utilizador novamente terá de ser reutilizada reestruturada para o seu estado anterior.

Figura 2: Ciclo de Vida de uma Aplicação



Fonte: celeiroandroid.blogspot.com/2011/02/ciclo-de-vida-de-uma-aplicão-android.html.

11.3. Monitoramento das atividades:

- O ciclo de vida completo de uma atividade acontece entre a primeira chamada do *onCreate (Bundle)* até o *onDestroy ().* Uma atividade vai fazer toda a configuração do estado geral no *onCreate ()* e liberar o recurso remanescente em *onDestroy*;
- A vida visível de uma atividade acontece entre o *onStart* até o *onStop*. Durante esse tempo o utilizador pode ver a atividade no ecrã pode não estar completamente visível ao utilizador as vezes;
- A vida no ecrã de uma atividade é completamente visível para o utilizador, acontece entre *onResume ()* até o *onPause ().* Durante esse tempo a atividade está na frente de outras atividades e interagindo com o utilizador, uma atividade pode ir frequentemente do *status* resume para *paused.*

12. Versões da plataforma Android

O Android ficou rapidamente conhecido por sua capacidade de personalização com atalhos, *Widgets* e papéis de parede animados.

Essas características mantêm até hoje e segue como principal diferencial em relação aos rivais, *iOS* da *Apple* e outros correntes tais como *Windows Phone* da *Microsoft*.

12.1. Android 1.0

Ainda sem nome, o Android foi lançado em 2008 *Smartphones G1*, que chegou somente aos EUA. O aparelho tinha ecrã sensível ao toque e um teclado físico retráctil para digitar. Daí que o Google começou a oferecer os seus serviços em versão móvel.

12.2. Android 1.5 Cupcake

A primeira versão conhecida por um nome de um doce, lançada em 2009. A utilização se destacou pelos atalhos da aplicação no ecrã inicial e a presença de pastas. O sistema também inovou ao permitir a adição de *Widgets* e *upload* de vídeos para YouTube e Picasa.

Figura 3: Android 1.5 Cupcake

Fonte: https://i0.wp.com/celular1.com.br/wp-content/uploads/2016/07/android-versoes-C%C3%B3pia-e1468767760223.png

12.3. Android 1.6 Donut

Trouxe em 2009 as primeiras aplicações da loja Android *Market* hoje conhecida como *Google play Store*. Também trouxe barras de pesquisas e compatibilidade com ecrãs com mais definições.

Figura 4: Android 1.6 Donut

Fonte: https://i0.wp.com/celular1.com.br/wp-content/uploads/2016/07/android-versoes-C%C3%B3pia-e1468767760223.png

12.4. Android 2.0 Eclair

Antes do fim de 2009 a Google lançou essa versão no *Smartphones Nexus One*. O sistema trouxe pela primeira vez a *Google Maps* com navegação via GPS, papéis de parede animados que permitiu adicionar mais página na ecrã inicial.

Figura 5: Android 2.0/2.1 Eclair

Fonte: https://i0.wp.com/celular1.com.br/wp-content/uploads/2016/07/android-versoes-C%C3%B3pia-e1468767760223.png

12.5. Android 2.2 Froyo

Chegou em 2010, com suporte ao *Adobe Flash*, *plugin*. Permitiu pela primeira vez partilhar internet móvel do aparelho por meio de cabo USB.

Figura 6: Android 2.2 Froyo

Fonte: https://i0.wp.com/celular1.com.br/wp-content/uploads/2016/07/android-versoes-C%C3%B3pia-e1468767760223.png

12.6. Android 2.3 Gingerbread

No final de 2010, chegou com um lançamento do *Smartphones Nexus 5*, fabricado pela Samsung. Trouxe partilha com chips NFC e sensores como acelerômetro e giroscópio, recursos que permitam a chegada de uma nova geração de *Apps* e jogos para *Smartphones*.

.

Figura 7: Android 2.3 Gingerbread

Fonte: https://i0.wp.com/celular1.com.br/wp-content/uploads/2016/07/android-versoes-C%C3%B3pia-e1468767760223.png

12.7. Android 3.0 Honeycomb

É a que menos impacto teve na história do *Software*. Foi feita exclusiva para *Tablets*, serviu como uma espécie de ponte para uma grande reformulação do sistema. A atualização foi importante para as novas inspirações para o novo *Design*.

Figura 8: Android 3.0 Honeycomb

Fonte: https://i0.wp.com/celular1.com.br/wp-content/uploads/2016/07/android-versoes-C%C3%B3pia-e1468767760223.png

12.8. Android 4.0 Ice Cream Sandwich

O Google lançou-o nos finais de 2011, com um Design profundamente modificado. A nova versão lançada em conjunto com *Smartphones Galaxy Nexus*. O sistema trouxe o botão exclusivo da aplicação recente.

Figura 9: Android 4.0 Ice Cream Sandwich

Fonte: https://i0.wp.com/celular1.com.br/wp-content/uploads/2016/07/android-versoes-C%C3%B3pia-e1468767760223.png

12.9. Android 4.1 Jelly Bean

O *Project Butter*, tecnologia que melhoraria o desempenho e a fluidez de animações. Lançada em 2012 com *Nexus* 4 da LG, com as atualizações levou o *Widgets* desenvolveu-se um ecrã de bloqueio, botões de ações e atalhos a área de notificação. O sistema permaneceu com o mesmo nome até a versão 4.3, foi a plataforma de introdução *do Google Now*.

Figura 10: Android 4.1 Jelly Beam

Fonte: https://i0.wp.com/celular1.com.br/wp-content/uploads/2016/07/android-versoes-C%C3%B3pia-e1468767760223.png

12.10. Android 4.4 KitKat

Lançado juntamente coma Nexus 5 em 2013, estreou-se a *Google Now* Launcher com atalho para *Google Now* a partir de um comando que desliza o ecrã inicial.

Figura 11: Android 4.4 KitKat

Fonte: https://i0.wp.com/celular1.com.br/wp-content/uploads/2016/07/android-versoes-C%C3%B3pia-e1468767760223.png

12.11. Android 5.0 Lollipop

Lançado em 2014, recebeu mais uma vez notificações relevantes no seu aspecto, com a chegada da linguagem Material de *Design*. Uma nova área de notificação e atalho.

Os telemóveis *Nexus* 6 e *Tablet Nexus* 9 foram os primeiros a oferecer suporte a dispositivo com arquitetura 64 bit.

Figura 12: Android 5.0 Lollipop

Fonte: https://i0.wp.com/celular1.com.br/wp-content/uploads/2016/07/android-versoes-C%C3%B3pia-e1468767760223.png

12.12. Android 6.0 Marshmallow

Em 2015 foi lançado a *Nexus 5X* e *Nexus 6P*. As atualizações trouxeram importantes recursos de segurança, controlo de permissões. Maior controlo de consumo de baterias em *stand bay*.

Figura 13: Android 6.0 Marshmallow

Fonte: https://i0.wp.com/celular1.com.br/wp-content/uploads/2016/07/android-versoes-C%C3%B3pia-e1468767760223.png

Versões mais recentes:

12.13. Android 7.0 Nougat

Foi lançada em 2016, essa versão começou a funcionar no primeiro Google Pixel. O *Software* passou a oferecer *Apps* em ecrãs divididas, notificações remodeladas e um atalho para aceder às aplicações recentes e mais rápidas.

Figura 14: Android 7.0 Nougat

Fonte: https://i0.wp.com/celular1.com.br/wp-content/uploads/2016/07/android-versoes-C%C3%B3pia-e1468767760223.png

12.14. Android 8.0 Oreo

Lançado em 2017, traz um sistema de gerenciamento mais avançado de baterias e funções de *Picture in Picture*, que cria janela flutuantes do YouTube. As atualizações contam com recursos *Autofull* para guardar palavra-chave e realizar login simplificado em *App*s e *Sites*. *Smart Text Selection* (seleção de textos inteligentes), usa inteligência artificial para reconhecer nomes de pessoas e estabelecimento para facilitar seleção e cópia de informações.

Figura 15: Android 8.0 Oreo

Fonte: http://imagens3.ne10.uol.com.br/blogsne10/mundobit/uploads//2017/08/OREO.jpg

13. Coordenadas Geográficas

As coordenadas geográficas são um sistema de linhas imaginário traçadas sobre o globo terrestre ou um mapa. É através da interação de meridiano com um paralelo que podemos localizar cada ponto da superfície da terra. Suas coordenadas a latitude e a longitude e o princípio utilizado é a graduação (graus, minutos e segundos).

Os paralelos e os meridianos são indicados por graus de circunstâncias. Um grau (1º) equivale a uma das 360 partes iguais em que a circunstância pode ser dividida. Um grau, por sua vez é igual a 59 minutos e 60 segundo.

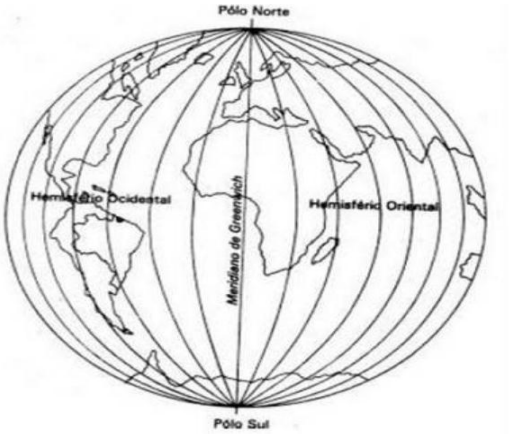
Os paralelos são linhas paralelas ao Equador, sendo que a própria linha imaginária do Equador é um paralelo. O 0º corresponde ao Equador, o 90º ao Polo Norte e o -90º ao Polo Sul.

Figura 16: Coordenadas Geográficas (Paralelos)

Fonte: www.sogeografia.com.br/Conteudos/GeografiaFisica/coodenadas.geo

Os meridianos são linhas perpendiculares ao Equador que vão ao Polo Norte, ao Polo Sul e cruzam o meridiano que passa pelo observatório de Greenwich, na Inglaterra. Logo o meridiano de *Greenwich* é o meridiano principal (0º). A Leste de Greenwich os meridianos são medidos por valores crescentes até 180º e a Oeste sua medidas são decrescentes até o limite de -180º.

Figura 17: Coordenadas Geográficas (Meridiano)



Fonte: www.sogeografia.com.br/Conteudos/GeografiaFisica/coodenadas.geo

A partir dos meridianos e paralelos, foram estabelecidas as coordenadas geográficas que são medidas em graus e a partir das coordenadas geográficas é possível localizar qualquer ponto da superfície da terra.

13.1. Latitude

É o ângulo formado entre o Equador e um ponto estimado. Todos os pontos do Equador possuem latitude geográfica igual a 0º. Pontos situados ao Norte do Equador têm latitude maiores que 0º, variando até 90º que é a latitude do Pólo geográfico Norte. Da mesma forma variam as latitudes ao Sul do Equador terrestre desde 0º ao 90º, latitude do Polo geográfico Sul. Para se diferenciar os valores, atribui-se sinal positivo para as latitudes Norte e negativo para as latitudes Sul.

13.2. Longitude

É o ângulo formado entre o meridiano que passa por determinado lugar e meridiano de Greenwich. A longitude é medida de 0º a 180º, para Leste ou para Oeste de Greenwich. Por convenção, atribui-se também sinais para as longitudes: Negativo para Oeste positivo para Leste. Aos termos os valores da latitude e da longitude de local desejado, teremos determinado as coordenadas geográficas do mesmo.

14. Computação móvel

Segundo os autores Carlos Maurício Seródio Figueiredo e Eduardo Nakamura (Computação móvel: Novas oportunidades e novos desafios, Junho de 2003), dos anos 90 até hoje, podemos destacar um grande crescimento no desenvolvimento de tecnologia para comunicação móvel, comunicação via satélite e redes locais sem fios.

A população dessas tecnologias tem permitido o acesso a informações remotas onde estiver, abrindo um leque muito grande de facilidades, aplicações e serviços para os utilizadores.

Computação móvel pode ser representada como um novo paradigma computacional que permite que o utilizador desses ambientes tenham acesso a serviços independentes de sua localização, podendo inclusive estar em movimento.

Um dispositivo para este fim deve ter a capacidade de realizar processamento, trocar informações via rede e ser capaz de ser transportado facilmente por seu utilizador. É importante que o dispositivo computacional tenha reduzido e não necessita de cabos para conectar a rede de dados ou fonte de energia eléctrica. Assim, equipamentos deste tipo devem ter as seguintes características:

- Ser bem menor que as estações de trabalho que são usadas frequentemente;
- Geralmente manipulado no colo ou na palma da mão;
- Possuir uma bateria, para evitar a necessidade de conexões a rede elétrica através de cabos que limitavam muito a mobilidade;
- Ter acesso a dados de tecnologia de redes sem fio.

CAPÍTULO III

15. Ferramentas e tecnologias utilizadas

15.1. Linguagem Java

Java é uma linguagem de programação orientada a objetos, desenvolvida por uma equipa de pessoas na *Sun Microsystems*.

Inicialmente elaborada para ser linguagem base de projetos de *Software* para produtos eletrônicos, *Java* teve seu grande impacto em 1995 devido ao sucesso mundial da *World Wide Web* (www).

A equipa da Sun desenvolveu um Browser totalmente escrito em Java, tendo-o terminado no início de 1995 e denominado de *Hotjava*. A grande diferença de *Hotjava* para outros *Browsers* da época tais como *Mosaic*, *Netscape* Navegador, e *Linux*, é que permitia inserção de programas escritos em *Java* dentro de páginas HTML comuns.

Hotjava como *Browser* não foi muito bom para a parte comercial, para os desenvolvedores as páginas HTML estaria a ser destinado a ser estático e sem ações embutidas em si, não houvesse uma linguagem padrão na qual fossem escritos programas que pudessem ser embutidas nas páginas Web. *Hotjava* demonstrou que isso era possível tinha de incluir programas, no caso escrito em *Java*, em uma única página HTML funcionando em um *Browser* preparado para suportar a execução do programa.

O grande impacto do Java foi quando *Netscape* anunciou uma versão do *Browser* navegador, iria dar suporte ás aplicações *Java* embutidas em documentos HTML.

Características e vantagens da linguagem Java**:**

- Orientação a Objeto;
- Portabilidade;
- Recursos de Rede;
- Segurança;
- Sintaxe similar a linguagem C/C++;
- Simplicidade na especificação;
- Carga dinâmica de código;
- É distribuído com um vasto conjunto de bibliotecas (*APIs*).

15.2. Linguagem JavaScript

Visando o potencial da internet para o público em geral e a necessidade de haver uma interação maior do utilizador com as páginas, a *Netscape*, criadora do navegador mais popular do início dos anos 90, de mesmo nome, criou o *Livescript*, uma linguagem simples que permitia a execução de *scripts* contidos nas páginas dentro do próprio navegador.

Apresentado o iminente sucesso do *Java*, que vinha conquistando cada vez mais espaço no mercado de desenvolvimento de aplicações corporativas, a *Netscape* logo realizou o *Livescape* como *JavaScript* num acordo com a Sun para alavancar o uso das duas. A então vice-líder dos navegadores, *Microsoft*, adicionou ao *Internet Explorer* o suporte a scripts escritos em *VBScript* e criou sua página própria versão de *JavaScript*, o *JScript*.

JavaScript é uma linguagem de programação mais popular no desenvolvimento Web. Suportando por todos os navegadores, a linguagem é responsável por praticamente qualquer tipo de dinamismo que queremos em nossas páginas.

Características de Linguagem *JavaScript*:

- O *JavaScript*, como o próprio nome sugere, é uma linguagem de *scripting*. Uma linguagem de *scripting* é comumente definida como linguagem de programação que permite ao programador controlar uma ou mais aplicações de terceiros. No caso do *JavaScript*, podemos controlar alguns comportamentos dos navegadores através de trechos de código que são enviados na página HTML;
- Outra característica comum nas linguagens de *scripting* são normalmente linguagens interpretadas, ou seja, não depende de compilação para serem executadas. Essa característica é presente no *JavaScript*: o código é interpretado e executado conforme é lindo pelo navegador, linha a linha, assim como o HTML;
- O *JavaScript* também possui grande tolerância a erros, uma vez que conversões automáticas são realizadas durante operações;
- O *scrip*t do programador é enviado com o HTML para o navegador, para que o navegador possa diferenciar o script de um código, é necessário envolver o *script* dentro da *tag* <*script*>.

15.3. Note.js

Note.js é uma plataforma do lado do servidor criado no *JavaScript Engine* (*V8 Engine*) do *Google Chrome* para criar facilmente aplicativos de rede rápidos e escalonáveis. O *Note.js* usa um modelo de E/S sem bloqueio orientado a eventos que o torna leve e eficiente, perfeito para aplicativos e tempo real que usam muitos dados e que são executados em dispositivos distribuídos. O *Note.js* foi desenvolvido por Ryan Dahi em 2009 e sua versão mais recente é v0.10.36. Também fornece uma rica biblioteca de vários módulos *JavaScript* que simplifica o desenvolvimento de aplicativos da Web usando o *Note.js* em grande medida.

Recursos do Note.js:

- **Assíncrono e baseado em eventos** – Todas as *APIs* da biblioteca *Note.js* são assíncronas, isto é, sem bloqueio. Essencialmente, significa que um servidor baseado em *Note.js* nunca espera por uma API para retornar dados. O servidor passa para a próxima após chamá-lo e um mecanismo de notificação de Eventos do *Note.js* ajuda o servidor a obter uma resposta da chamada da API anterior;
- **Muito rápido** – Sendo construída no mecanismo *JavaScript V8* do *Google Chrome*, a biblioteca *Note.js* é muito rápido na execução de código;
- **Single Threaded, mas altamente Escalável –** O *Note.js* usa um único modelo encadeado com *loop* de eventos. O mecanismo de eventos ajuda o servidor a responder de maneira não-bloqueante e torna o servidor altamente escalável, ao contrário dos servidores tradicionais, que criam encadeamento limitados para lidar com solicitações. O *Note.js* usa um único programa de encadeado e o mesmo programa pode fornecer serviço para um número muito maior de solicitações do que os servidores tradicionais, como o *Apache HTTP Serve;*.
- **Sem armazenamento em Buffer –** os aplicativos *Note.js* nunca armazena nenhum dado em *buffer*. Esses aplicativos simplesmente exibem dados em partes;
- **Licença –** *Note.js* é liberada sob a licença MIT.

15.4. Linguagem HTML

Segundo Yuri Pacievitch HTML é uma linguagem de marcação utilizada para desenvolvimento de *Sites*. Surgiu com o HTTP.

Foi criada em 1991 por *Berners-Lec*, CERN (*European Council for Nuclear Research*) na Suíça. Inicialmente o HTML foi projetado para utilizar instruções de pesquisa próximas e compartilhar documentos com facilidade. Em 1992 foi disponibilizado a biblioteca de desenvolvimento www (*world wide web*), uma rede de alcance mundial.

Essa linguagem de marcação é constituída por código que delimitam conteúdo específico, possui código para criação de páginas na Web.

Estes códigos definem o tipo de letra, tamanho, cor, espaçamento e vários outros aspectos de *Sites*.

Foi a primeira linguagem a nível mundial, porém não é a única, existem muitas criações de páginas Web. Atualmente já é possível integrar várias linguagens na mesma página Web, sendo possível usar duas ou mais linguagem no mesmo *Site*.

Para criar e editar código HTML é necessário qualquer editor de texto comum, como bloco de notas. Para testar os códigos, basta guardar o arquivo em formato ".HTML" e executar.

Para o teste é necessário ter um navegador configurado como padrão, não á necessidade de internet.

Alguns códigos em HTML e suas funções:

- **<title>** - Define o título da página;
- **<script>** - Permite adicionar funções em páginas com *script*, podendo assim adicionar código em *JavaScript* (este código permite que alguns sites em HTML ter jogos e animações de formulário antes do envio para o servidor entre outras funcionalidades;
- **<style>** - define formatação em CSS.

15.5. Linguagem CSS

CSS (*Cascating Style Sheets*) é uma "folha de estilo " composta por "camadas" para definir a apresentação (aparência em páginas da internet que adotam para o seu desenvolvimento, linguagem marcação (como XML e HTML). CSS define como

serão exibidas os elementos contidos no código de uma página da internet e a sua maior vantagem é efetuar a separação entre formato e o conteúdo de um documento.

Com a evolução dos recursos de programação as páginas da internet estavam adotando cada vez mais estilos e variações para deixá-los mais elegantes e atrativos para os utilizadores. Com isto a linguagem de marcação simples do HTML, que era destinada para apresentar os conteúdos também precisou ser aprimorada.

Foram criadas novas Tags e atributos de estilo para HTML e em resumo, passou a exercer tanto a função de estruturar o conteúdo quando apresenta-lo para o utilizador final. Entretanto o começou a trazer problemas para os programadores, pois não havia forma de definir padrões para estilos ou cabeçalhos ou conteúdos em diversas páginas, ou seja as alterações tinha que ser feitas uma a uma.

Algumas vantagens do CSS:

- Possibilidade de controlo do Layout vários documentos a partir de um único arquiva CSS;
- Possibilidade de manter a mesma formatação em diferentes navegadores;
- Aplicação de técnicas mais sofisticadas de desenvolvimento;
- Menor consumo de banda e melhor desempenho devido ao reuso do mesmo código de formatação em vária páginas.

CSS3

É simplesmente uma versão mais recente da CSS. Incorpora novos elementos para construir animações tanto em 2D e 3D. Incorpora também novos mecanismos para maior controlo sobre estilo com o qual se mostram as características das páginas.

15.6. JSON

JSON é um formato de dados baseado em texto que segue a sintaxe do objeto *JavaScript* que foi popularizado por *Douglas Crockford*.

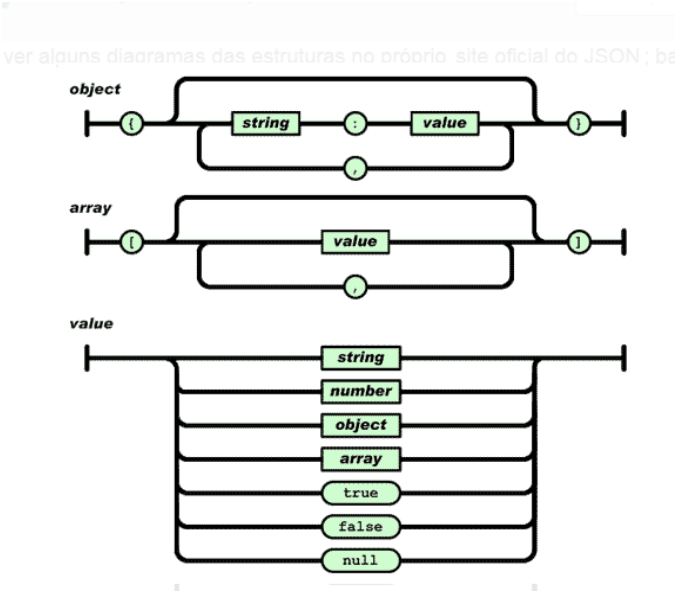
Pode ser usada independentemente do *JavaScript* e muitos ambientes de programação apresentam a capacidade de ler (analisar) e gerar JSON.

JSON existe como uma *"string"* é útil quando deseja transmitir dados por uma rede. Precisa de ser convertida em um aspecto *JavaScript* nativo quando

deseja acessar os dados. O JSON pode realmente assumir a forma de qualquer tipo de dados que seja válido para inclusão dentro dele, não apenas matrizes ou objetos

Ao contrário do JavaScript no qual as propriedades do objeto podem estar sem aspas, em JSON somente aspas podem ser usadas como propriedades.

Figura 18: Estrutura do JSON



Fonte: http://desenvolvimentoparaweb.com/javascript/json-javascript-object-notation/2

15.7. Android Studio

Android Studio é um ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicações Android é baseado no *intelliJ IDEA.*

Além do editor de código e das ferramentas de desenvolvedor avançados de intelliJ, oferece ainda mais recursos para aumentar sua produtividade na criação de aplicações Android como:

- Um sistema de compilação flexível baseado no *Gradle*;
- Um emulador rápido com inúmeros recursos;
- Um ambiente unificado para desenvolver para todos os dispositivos Android;
- Instant Run para aplicar alterações a aplicações em execução sem precisar compilar um novo APK;
- Compatibilidade com C++ e NDK;
- Ferramentas de verificação de código suspeitam para detectarem problemas de desenvolvimento, usabilidade, compatibilidade com versões e outros.

15.8. Estrutura do Projeto no Android Studio

Cada projeto no Android Studio conte um ou mais módulos com arquivos de código fonte e recursos.

Os tipos de módulos incluem:

- Módulo de aplicação Android;
- Módulo de biblioteca;
- Módulo de Google App Engine.

Todos os arquivos de compilação podem ser vistos no nível superior em *Gradle* scripts e cada módulo da aplicação contém as seguintes partes:

- Manifest: contém o arquivo *AndroidManifest.xml*;
- Java: contém arquivos de código fonte Java, incluindo o código de teste da *JUnit*;
- Recursos: contém todos os recursos que não são código *Layouts XML*, *string* de UI e imagens em *bitmap*.

15.9. Android SDK

Android SDK ou *Kit* de Desenvolvimento de Software para Android é um pacote com diversas ferramentas utilizadas pelo Android Studio e pelos desenvolvedores Android, incluindo componentes como SDK *tools*, *Build tools* e a *Plataforma tools*.

Permite aos desenvolvedores criarem aplicações para plataforma Android de forma nativa. Android SDK inclui projetos de exemplo com código fonte, ferramentas de desenvolvimento, emuladores e bibliotecas necessários para criar as aplicações Android.

As aplicações são escritas usando a linguagem de programação *Java* e executados na *ART Delv*ik, máquinas virtuais personalizadas e projetadas para funcionar dentro dos dispositivos Android que funciona em cima de um *karnel Linux*.

A ferramenta SDK pode ser usada por linha de comando, o método é usar um ambiente de desenvolvimento integrado (IDE). O IDE é recomendado é Android Studio, ferramenta oficial do Google para desenvolvimento Android.

15.10. Visual Paradigm

Visual Paradigm é um pacote de desenvolvimento de *Software* e gerenciamento corporativo, que fornece todos os recursos para arquitetura corporativa, que fornecedor dos recursos para arquitetura corporativa, gerenciamento de projetos, desenvolvimento de *Software*.

Funcionalidades do Visual Paradigm:

- Ferramentas UML e SysML;
- Ferramentas e diagramas BPMN;
- UML para código, código para UML;
- Ferramenta de gerenciamento de projetos;
- Formatos de saída versáteis;
- Ferramentas de geração de documentos;
- Guia de ciclo de vida de gerenciamento de projeto;

15.11. Notepad++

Notepad++ é um editor de código fonte, um substituto do *Notpad* que suporta vários idiomas no ambiente *MS Windows* ou uso é rápido pela licença GPL.

Baseado no poderoso componente de edição *Scintilla*, O Notepad++ é escrito em C++ e usa API Win32 puro e STL, o que garante uma maior velocidade de uso.

Características:

- *Realce* de *sintaxe* e dobra de *sintaxe*;
- Multi Documentos (interface de guia);
- Visão múltipla;
- Ambiente multilingue suportado;
- Gravação e reprodução de macro;
- Lançar com argumentos.

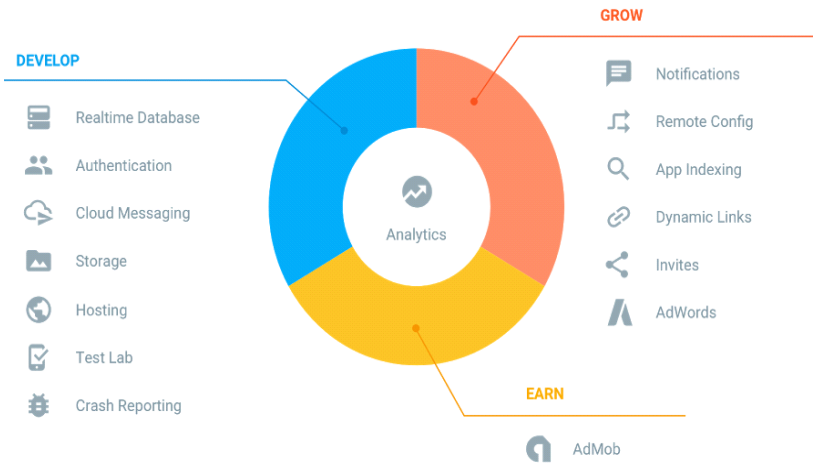
15.12. Firebase

Segundo Carlos Gasperin, Firebase é uma plataforma de desenvolvimento mobile e Web adquirida pela Google em 2014.

Com certeza em ser um *back-end* completo e de fácil usabilidade, essa ferramenta disponibiliza diversos serviços diferentes que auxiliam no desenvolvimento e gerenciamento de aplicações.

Para utilizar o *Firebase*, uma console Web foi criado para facilitar a implementação. Neste caso o desenvolvedor adiciona um projeto e inclui os serviços que desejar, cada um com uma explicação de como proceder. Nem todos os serviços são gratuitos, porem é possível criar planos conforme as necessidades do desenvolvedor.

Figura 19: Estrutura do Firebase



Fonte: www.mecreiros.com/firebase-o-que-e-e-como-funciona

Serviços do Firebase:

- *Realtime Database;*
- *Authentication;*
- *Cloud Messaging;*
- *Storage;*
- *Hosting;*
- *Test Lab;*
- *Crach Reporting;*
- *Notifications;*
- *Remote Config;*

- *App indexing;*
- *Dynamic Links;*
- *Invites;*
- *Ad Words;*
- *Ad Mob;*
- *Analytcs*.

15.13. Firebase Realtime Database

Firebase Realtime Database é uma base de dados hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real com todos os utilizadores conectados. Quando cria um App em plataformas cruzadas com SDKs para *iOS,* Android e *JavaScript* todos os utilizadores compartilham uma instância do *Realtime Database* e recebem automaticamente atualizações com dados mais recentes e anda os dados permanecem disponíveis quando o *App* está *off-line*.

Realtime Database fornece uma linguagem de regras flexíveis baseado em expressões, denominadas de regras de segurança para definir como os dados são estruturados e quando. Por meio de integração com *Firebase Autentication*, os desenvolvedores podem definir quem tem acesso, a quais dados e como esses dados poder ser acessados.

É uma base de dados *NoSQL* e, por isso tem otimizações e funcionalidades diferentes de uma base de dados relacional. A API do *Realtime Database* autoriza operações que possam ser executadas com rapidez. Isso possibilita uma óptima experiência em tempo real que atende milhões de utilizadores sem permitir a capacidade de resposta.

Principais recursos:

- **Em tempo real** – Usa a sincronização de dados sempre que os dados são alterados, todos os dispositivo conectados recebem essa atualização em milissegundos. Cria experiência colaborativa e imersivas sem se preocupar com código de rede;
- **Off-line** – Os *Apps* de *Firebase* permanecem responsivo mesmo *off-line*, pois o SDK do *Firebase Realtime Database* mantem seus dados em disco. Quando a conectividade é restabelecida, o dispositivo do cliente recebe as alterações perdidas e faz a sincronização com o estado atual do servidor;

- **Acessível** - em dispositivo do cliente – pode ser passado diretamente de dispositivo móvel ou navegador da Web, sem um servidor de aplicações. A segurança e validação de dados estão disponíveis por meio de regras de segurança baseada em expressão de *Firebase Realtime Database*, executando quando os dados lidos ou gravados;
- **Escalonar em vária base de dados** – com *Firebase Realtime Database* no plano de preços de *Blaze*, pode oferecer suporte em grande escala as necessidades da *App*.

Pode dividir os dados entre várias instâncias de base de dados no mesmo projeto do *Firebase*.

Simplifica a autenticação no seu projeto com o *Firebase Authentication* e autentique utilizadores na sua instância de base de dados.

15.14. Firebase Authentication

A maioria das *Apps* precisa reconhecer a identidade do utilizador. Ter essa informação permite que um *App* guarde os dados do utilizador na nuvem com segurança a e fornece a mesma experiência personalizada em todos os dispositivos do utilizador.

Firebase Authentication fornece serviços de *Back-end*, SDKs fáceis de usar e bibliotecas de IU prontas para autenticar utilizadores no seu *App*. Oferece suporte a autenticação por meio de palavra-chave, números de telefone e provedores de identidade como *Google, Facebook, Twitter* e muito mais.

É estritamente integrado a outros serviços do *Firebase* e aproveita os padrões de sector, como *OAuth 2.0* e *OpenID Connect* para possa ser facilmente integrado ao seu *back-end* personalizado.

Para conectar um utilizador ao seu *App*, precisa ter as credenciais de autenticação do utilizador. Essas credenciais podem ser o endereço de Email e a senha do utilizador ou um *Token* do *OAuth* de um provedor de identidade federado. Transmitir essas credenciais para SDK do *Firebase Authentication*. Os serviços *back-end* verificam essas credenciais e enviará uma resposta ao cliente.

Principais recursos:

Pode permitir que o utilizador faça *login* no *App* de *Firebase* usando o *Firebase UI* como uma solução de autenticação *drop-in* completa ou SDK do Firebase *Authentication* para integrar um ou vários métodos de login no *App*.

Firebase UI Auth:

- Solução de autenticação com *drop-in*;

SDK do *Firebase Authentication*:

- Autenticação baseada em Email e senha;
- Integração do provedor de identidade federado;
- Autenticação por número de telefone;
- Integração de sistema autenticação personalizada;
- Autenticação anónima.

15.15. Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) é uma forma de enviar mensagem entre plataformas que permite a entrega confiável de mensagem sem nenhum custo.

Usando SDK pode notificar um *App* cliente de que novos *emails* ou outros dados estão disponíveis para sincronização. Podem enviar mensagem de notificações para promover novas alterações e a retenção de utilizador para caso de uso como mensagens instantâneas, uma mensagem pode transferir um *payload* de até 4 KB para um App cliente.

Principais recursos:

- Enviar mensagens de notificação ou mensagens de dados;
- Segmentação versátil de mensagens;
- Enviar mensagens das Apps dos clientes.

Uma implementação do FCM inclui dois componentes principais para envio e recebimento:

- Um ambiente confiável como *Cloud Function* para *Firebase* ou um servidor de App que é usado para criar, segmentar e enviar mensagens;
- Um App cliente iOS, Android ou Web (JavaScript), que recebe as mensagens.

15.16. Firebase Hosting

Firebase Hosting é uma hospedagem de conteúdos Web de nível de produção para desenvolvedores. Usando *Hosting*, implante App da Web e conteúdos estático de modo rápido e fácil em uma rede de distribuição de conteúdos (CDN) global com um único comando.

Foi criado pensando no desenvolvedor Web moderno. Os *Sites* estáticos estão mais avançados com surgimento das estruturas de *front-end* de *JavaScript*, tais como ferramentas de gerador angular é estático.

Além da hospedagem de conteúdo estático, o *Firebase Hostin*g conta com opções de configuração leves para que possa criar *Progressive Web* e *App* sofisticado. É possível reescrever URLs para o roteamento do cliente ou configurar cabeçalhos personalizados com facilidade.

Principais recursos:

- Servidor por uma conexão segura;
- Entrega rápida de conteúdos;
- Implantação rápida;
- Rollback de um cliente.

15.17. Notification

È possível gerenciar campanhas de notificação para seu *App* e quando integrado ao *Firebase Analytics* é possível entregar mensagens a um segmento específico de utilizador.

15.18. AngularJS

AngularJS é um *Framework* em JavaScript, de código aberto e que é mantido pelo *Google*.

Pode ser acessado por um navegador Web e tem como padrão o MVVM (*Model-View-View-Model*).

O *Framework* AngularJS funciona através de leitura de páginas HTML, tem como âmbito atributos adicionais personalizados em suas *tags*. Angular interpreta esses dados como as diretivas para ligar partes de entrada ou saída de páginas para um modelo que é representado por variáveis em padrão *JavaScript*. Os valores dessas variáveis *JavaScript* podem ser configurados manualmente no código ou recuperação a partir de recursos JSON estáticos ou dinâmicos.

O *AngularJS* disponibiliza recursos completos para facilitar a criação de uma aplicação CRUD:

- Vinculação de dados;
- Directores baseados em métodos;
- Validação de formulários;
- Roteamento;

- Componentes reutilizáveis.

Características e Recursos de AngularJS:

- Utilizar padrão MVC (*Model View Controller*);
- É baseado no conceito SPA (*Single Page Application*);
- Utiliza o conceito de Injeção de Dependência;
- Utilizar recurso *Two-Way Data Binding*;
- Utiliza recursos de Diretivas.

É distribuído como um arquivo *JavaScript* e pode ser adicionado a uma página da Web com uma *tag* de *script*.

Figura 20: Exemplo de como è estruturado do AngularJS no projeto

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.0/angular.min.js"></script>
```

Fonte: Elaboração Própria

CAPÍTULO IV

16. Análise do sistema a desenvolver

Nesta parte do meu Relatório irei falar sobre os requisitos do sistema e a modelagem através dos gráficos e diagramas.

16.1. Visão geral do sistema

Tendo em conta o desenvolvimento do protótipo, este dividido em duas partes:

Na parte de Aplicação – O utilizador vai ter uma intercessão com os serviços, onde escolhido quais serviços pretende escolher e os procedimentos necessário que deve seguir, para transmitir informações que serão armazenadas numa base de dados.

De um modo geral nesse sistema o utilizador tem que fazer um login onde seus dados serão juntamente associadas as outras informações que serão disponibilizadas mais adiante.

Na parte Web – A entidade responsável vai fazer a consulta dos dados, verificar se todos os dados necessários chegaram, para dar sequência aos pedidos solicitados. Antes disso a entidade tem que fazer um *login* para poder ter acesso a esses dados e dar continuidade ao procedimento necessários.

Esses dados são atualizados automaticamente devido a plataforma *Firebase* onde utiliza o *Realtime Database*.

17. Análise dos Requisitos do Sistema

17.1. Requisitos funcionais para Aplicação

- **Sistema de login** – O utilizador tem que fazer uma autenticação tanto pelo *Gmail*, mero de telefone, ou outro *Email* qualquer;
- **Apresentação de um Menu** – o utilizador vai ter opções de serviços onde pode escolher;
- **Criação de um perfil** – O utilizador preenche um pequeno formulário onde é deixado informações básicas;
- **Consulta de Mensagens** - Consulta de mensagem que é enviado pela que vai prestar socorro;
- **Serviços** – Vai ter um conjunto de opções de envio dentro de cada serviço;
- **Localização** – O utilizador disponibilizar a posição atual para uma entidade.

17.2. Requisitos funcionais para página Web

- Sistema login – A entidade tem que autenticar de modo a ter acesso às informações;
- Apresentação de um Menu – A entidade vai ter acesso às informações do utilizador;
- Apresentação das informações dos utilizadores – A entidade vai ter acesso aos tipos de ocorrência e os dados do perfil do utilizador;
- Mensagens – A entidade pode enviar um SMS para o utilizador caso for necessário;
- Localização – A entidade pode obter a posição atual através de pontos no mapa.

17.3. Requisitos não funcionais

- Disponibilidade de conexão com internet, em caso de falha na rede os dados são atualizados automaticamente, depois que a rede voltar a normalidade o sistema do *Realtime Database* da *Firebase* atualiza os dados;
- Pode ser instalado em qualquer dispositivo móvel;
- Utilização de linguagem *Java* que é mais compatível para dispositivos Android, e para leitura dos dados no servidor a linguagem utilizada é JSON, HTML e CSS pela parte Web;
- Pouco consumo da bateria o protótipo não exige muito do dispositivo;
- Tanto na parte do *App* e na parte Web vai ter uma boa visibilidade ou seja o utilizador e a entidade podem fazer o manuseamento dos conteúdos sem nenhum problema, as opções estão bem nítidas.

18. Modelação do Sistema

Segundo os Autores Aberto Silva e Carlos Vieira (UML, Metodologia e Ferramentas CASE, 1º Edição Abril 2011, Pág. 111 e 120), UML (*Unified Modelling Language*) é uma língua diagramática, utilizável para especificação, visualização de documentação de sistemas de Software. O UML surgiu em 1997 na sequência de um esforço de unificação de três principais linguagens de modelação orientadas por objeto (OMT, Booch, OOSE). Adquiriu o estrato de norma no âmbito de OMG e a ISO tendo vindo a ser adaptado progressivamente pela indústria e academia em todo mundo.

O UML apresenta, entre outras as seguintes características principais:

- È independente do domínio de aplicação (pode ser usado em projetos de diferentes características como: sistema clientes/servidor tradicionais, sistema baseado na Web, sistema de informação geográfica e sistema de tempo real;
- É independente do processo ou metodologia de desenvolvimento;
- É independente das ferramentas de modelação;
- Apresenta mecanismos potentes de extensão;
- Agrega um conjunto muito significativo de diferentes diagramas /técnicos dispersos por diferentes linguagens (diagrama de caso de utilização, de classes, de objectos, de colaboração, de atividade, de estado, de componentes, e de instalação).

18.1. Diagrama de caso de utilização

Segundo Mauro Nunes e Henriques O'Neill (Fundamental de UML, 4º edição, 2004, Pág. 13 e 14), os use case, ou casos de utilização constituem a técnica em UML para representar o levantamento de requisitos no desenvolvimento de sistemas de informação tenta garantir que o sistema seja útil para o utilizador final estando de acordo com suas necessidades.

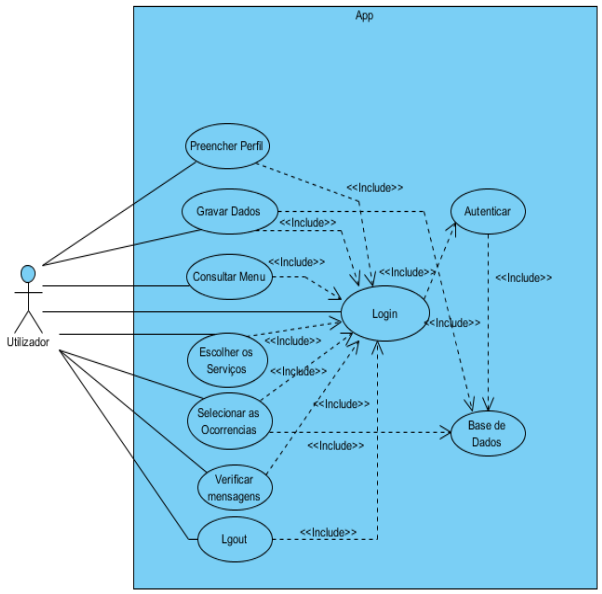
Os diagramas de use case são utilizados para representação de requisito e para assegurar que tanto o utilizador final como o perito numa determinada área ou o especialista. O seu objetivo é mostrar que um sistema deve efetuar e não como o vai fazer.

Esses diagramas utilizam os seguintes abstrações de modelação:

- Atores;
- Use case;
- Relações (<<*include*>>, <<*extend*>> e Generalização).

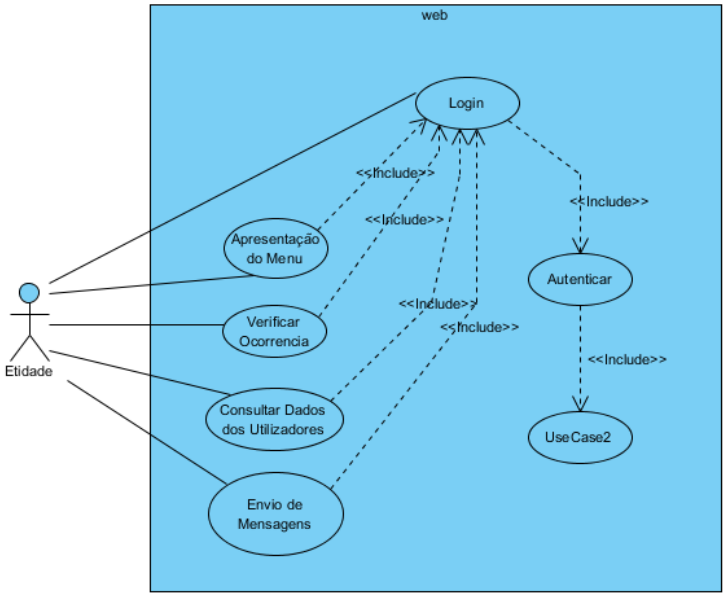
Nas figuras 21 e 22 abaixo mostram os respetivos Diagramas de caso de Utilização, tanto na parte do App como na parte Web.

Figura 21: Diagrama de caso de Utilização do Sistema (Aplicação)



Fonte: Elaboração Própria

Figura 22: Diagrama de caso de Utilização do Sistema (Web)



Fonte: Elaboração Própria

18.2. Diagrama de Classe

Segundo Mauro Nunes e Henriques O'Neill (Fundamental de UML, 4º edição, 2004, Pág. 35 e 36), a UML optou também o diagrama de classes, uma das técnicas mais utilizadas no desenvolvimento orientado por objeto. Este diagrama é uma discrição forma de estrutura de objectos num sistema. Para cada objecto descreve a sua identidade, os seus relacionamentos com os outros com objetos os seus atributos e as suas operações.

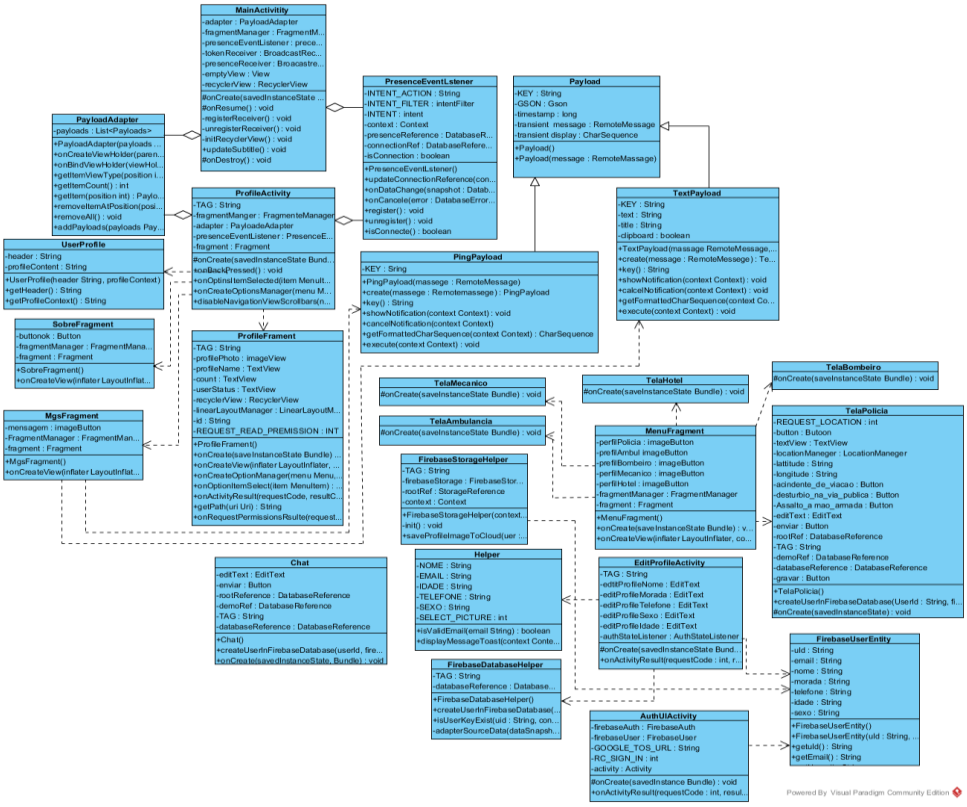
Segundo Mauro Nunes e Henriques O'Neill (Fundamental de UML, 4º edição, 2004, Pág. 35 e 36), a criação de um modelo de classes resulta de um processo de abstração através do qual se identificam os objectivos (entidades e conceitos) relevantes no contexto que se pretende modelar e se procuram descrever características comuns em termos de descrição genérica designa-se por classe. Assim, as Classes descrevem dois propósitos: permitem compreender o mundo real naquilo que é relevante para o sistema de informação que se pretende desenvolver e fornecem uma base pratica para a implementação em computador (*Rumbaugh* et al, 1991).

Um diagrama de classe é composto pelos seguintes elementos abstractos de modelação:

- Classes de Objetos;
- Relações de associação e Generalização;
- Multiplicidade.

Na figura 23 abaixo mostra o respetivo Diagramas de Classe do Sistema desenvolvido.

Figura 23:Diagrama de Classe do Sistema



Fonte: Elaboração Própria

18.3. Diagrama de Actividade

Segundo Mauro Nunes e Henriques O'Neill (Fundamental de UML, 4º edição, 2004, Pág. 57,58 e 59), o diagrama de actividade constitui um elemento de modelação simples, mas eficaz para descrever fluxos de trabalho numa organização ou para detalhar operações de uma classe, incluindo comportamento que possuem processamento paralelo.

Segundo Mauro Nunes e Henriques O'Neill (Fundamental de UML, 4º edição, 2004, Pág. 57,58 e 59), no contexto dos sistema de informação da gestão, define-se processo de negócio com um conjunto integrado de atividade de uma organização, que procura satisfazer um determinado objetivo e no qual participam um ou mais autores.

Como já foi referido anteriormente, um *use case* pode ser utilizado para identificar um processo de negócio de uma organização. O diagrama de atividades é assim particularmente útil quando pretende detalhar um *use case* associado a um processo de negócio.

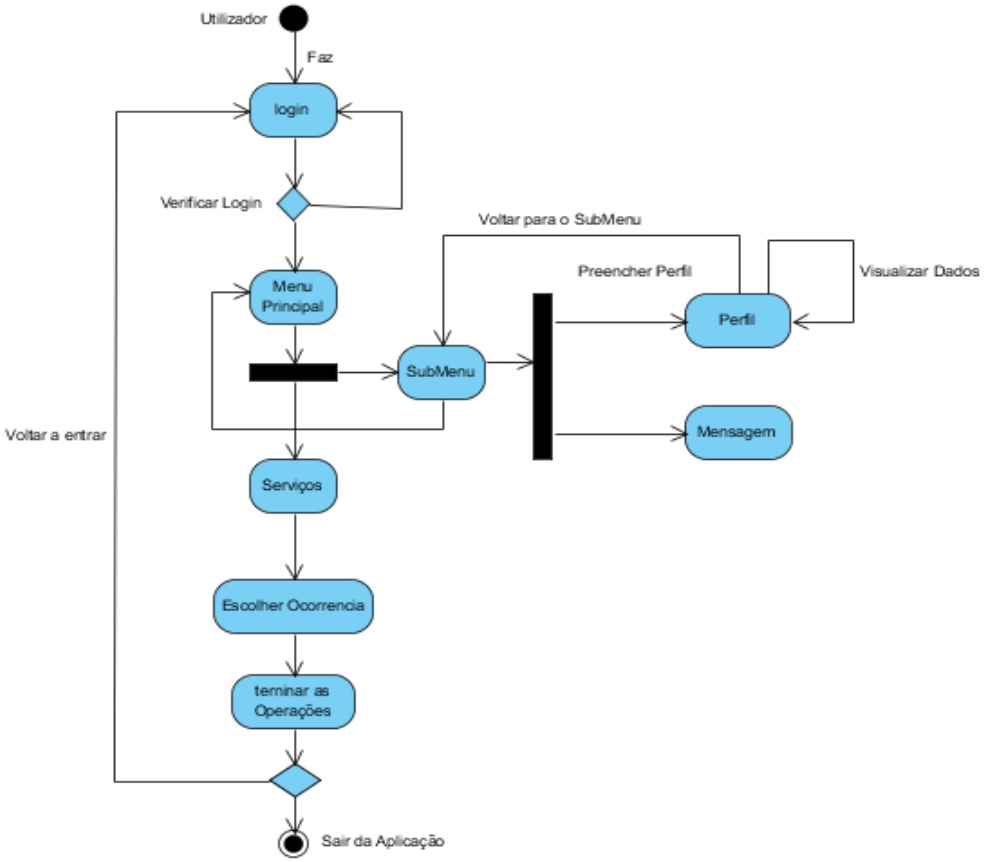
Um diagrama de atividades pode ainda ser utilizado na descrição de um fluxo de atividades mais alargados, envolvendo diversos usem case.

No diagrama são utilizados os seguintes elementos de modelação:

- Linhas verticais de responsabilidades;
- Actividade de Início e de Fim;
- Actividade intermédias;
- Transição de atividades e símbolos de comportamento condicional.

Na figura 24 abaixo mostra o respetivo Diagramas de Actividade do Sistema desenvolvido.

Figura 24: Diagrama de Actividade do Sistema



Fonte: Elaboração Própria.

18.4. Diagrama de sequência

Segundo Alberto Silva e Carlos Videira (UML, Metodologia e Ferramentas CASE, 2º edição, Volume 1, Pág. 198 e 199), um diagrama de sequência ilustra uma interação segundo uma visão temporal.

Um diagrama de sequência é representado através de duas dimensões: a dimensão horizontal, que representa um conjunto de objetos intervenientes; e a dimensão vertical que representa o tempo. A apresentação desta dimensão pode ser invertida, se for conveniente.

Não existe qualquer significado na ordenação horizontal dos objectos intervenientes, ou seja, na sua disposição relativa.

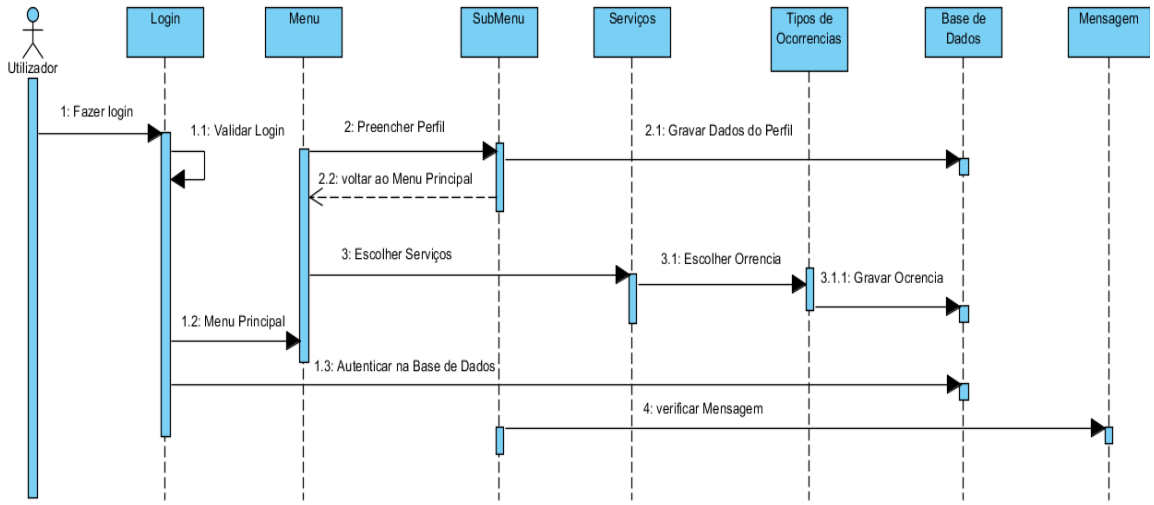
Nos diagramas de sequência as setas são desenhadas horizontalmente de forma a representar a indivisibilidade de ações correspondentes ao envio da mensagem.

Segundo Mauro Nunes e Henrique O'Neill (Fundamental de UML, 4º edição, 2004, Pág. 78), diagrama de sequência é composto pelos seguintes elementos abstractos de modelação:

- Objectos;
- Ligação (*links*);
- Mensagem.

Na figura 25 abaixo mostra o respetivo Diagrama de Sequência do Sistema desenvolvido.

Figura 25: Diagrama de Sequência do Sistema



Fonte: Elaboração Própria.

18.5. Diagrama de classe de Entidade Relacionamento

Segundo os autores Filomena Castro, Maria Paula Morais e Armando Jorge (Desenvolvimento de Sistema de Informação, Fevereiro 2005, Pág. 110), o diagrama entidade relacionamento (E-R) representa, tal como o nome indica, as entidades envolvidas no SI em estudo, bem como as relações que existe entre essas entidades. Faz uma representação estática do sistema, mostrando somente os seguintes objetos: entidade e relações.

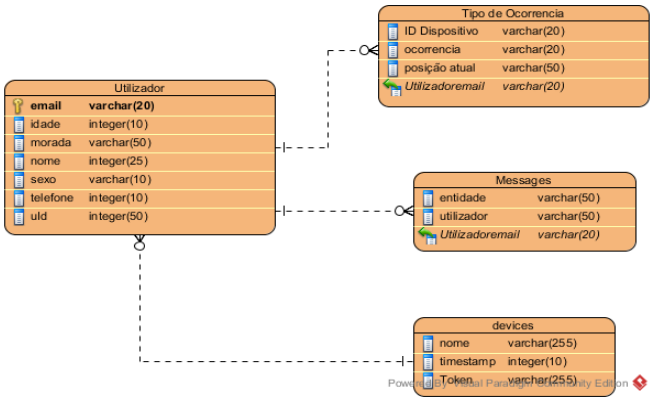
. Os elementos que constituem um diagrama são:

- Entidade – qualquer coisa ou abstracta de importância para o sistema em estudo e sobre a qual se tem que guardar;
- Relação – qualquer tipo possível de ligação que possa existir entre as entidades;
- Grau de associações - mostra como estão ligados quantitativamente duas entidades relacionais.

A forma com os autores representam a entidade, a relação, o grau de associação entre as entidades pode definir. Há autores que se limitam a representar o nome da entidade, outros identificam também atributos que caracterizam essas entidades. No entanto o símbolo para representar uma entidade é sempre o mesmo – o rectângulo. As principais diferenças de notação que se encontra nos diferentes diagramas E-R estão na forma de representar a relação, sobre uma linha ou dentro de um losango.

Na figura 26 abaixo mostra o respetivo Diagrama de Classe de Entidade Relacionamento do Sistema desenvolvido.

Figura 26: Diagrama de Classe de Entidade Relacionamento do Sistema



Fonte: Elaboração Própria

19. Protótipo

É uma versão inicial, reduzida proporcionalmente da solução de sistema ou de parte de uma solução de sistema construído em um curto período de tempo e aprimorada em várias iterações para testar e avaliar a eficácia de *design* global utilizado para resolver problemas específicos.

O protótipo são usados de uma forma direta para reduzir o risco, pode reduzir a incerteza sobre:

- A viabilidade de negócio de um produto que está sendo desenvolvido;
- A estabilidade ou o desempenho da tecnologia chave;
- Compromisso do projeto ou precisão de fundos, construindo um pequeno protótipo de prova de conceito;
- O atendimento de requisitos;
- A aparência do produto, sua aparência.

Tipos de Protótipo

- **Protótipo Exploratórios** – é projetado para ser parecido com um pequeno experimento para testar algumas principais premissas sobre o projeto, funcionalidades, tecnologia. Ele pode ser tão pequeno quanto algumas centenas de linha de código, criadas para testar o desenvolvimento de um Software chave ou componentes de Hardware, ou pode se uma maneira de esclarecer os requisitos;
- **Protótipos Evolutivos** – evoluem de uma iteração para o próximo. Enquanto houver qualidade de produção inicialmente, o código tende a ser retalhado conforme o produto é desenvolvido. Para gerenciar o trabalho, tendem a ser projetados mais formalmente e testado com uma certa formalidade mesmo nos estágios iniciais;
- **Protótipos Comportamentais** – Tendem a ser protótipos Exploratórios, não tentam reproduzir a arquitectura do sistema a ser desenvolvido mas em vez disso, tem foco em que o sistema terá quando visto pelos utilizadores.
- **Protótipos Estruturais** - tendem a ser evolutivos, são mais susceptíveis a utilização da infraestrutura do sistema final e provavelmente serão desenvolvidos para se tornarem o verdadeiro sistema.

Vantagens no uso de Protótipos:

- Facilitam o entendimento e o *feedback* dos utilizadores;

- Cumprem o desejo de mostrar resultados rápidos para o cliente;
- Tornam as discussões mais produtivas e sob controlo na sessão com os utilizadores.

20. Arquivo AndroidManifest.xml no Android

O arquivo *AndroidManifest* conte informações do pacote incluindo componentes da aplicação, como *activity*, *service*, *broadcast reservers*, *content provaider* etc.

Tarefas que são executadas:

- É responsável por proteger a aplicação para acessar qualquer parte protegido, fornecendo as permissões;
- Declara a API do Android que a aplicação vai usar;
- Faz lista as classes de instrumentação. As classes de instrumentação fornecem perfis e outras informações, estas informações são removidas imediatamente antes da publicação da candidatura, etc.

Elementos do arquivo Androidmanifest.xml:

<manifest> é o elemento raiz do arquivo *AndroidManifest.xml*, tem um atributo de pacote que descreve o nome do pacote da classe de actividades;

<application> é o sub elemento do manifesto inclui a deslocação de *namespace*. Esse elemento contém vários sub elementos que declaram o componente de aplicação, como atividade etc.

Os atributos comumente usados são desse elemento: *icon, label, theme*:

- **Android: icon** – representa o ícone para todos os componentes da aplicação Android;
- **Android: label** – funciona como rótulo padrão para todos os componentes da aplicação;
- **Android: theme** - representa um tema comum para todas as atividades do Androide.

<**activity**> é o subelemento da aplicação e representa uma atividade que deve ser divida no arquivo *AndroidManifest.xml*, tem muitos atributos como rótulo, nome, tema *launch Mode* :

- **Android: label** – representa um rótulo, ou seja, exibi-lo na tela;

- **Android: name** – representa um nome para classe de **atividade**. É atributo obrigatório.

<**intent-filter**> é o subelemento de atividade que desenvolve o tipo de intenção ao qual a atividade, o serviço ou o receptor de broadcast podem responder.

<**action**> Adiciona uma ação para o *intent-filter* . O *intent-filter* deve ser pelo menos um elemento de ação.

<**category**> Adiciona um nome de categoria a um *intent-filter*.

Na figura abaixo mostra a constituição do AndroidManifest do projeto a ser desenvolvido

Figura 27: Arquivo AndroidManifest.xml do Sistema

```
    package="com.kelvin.miapp">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" /

    <application
        android:name=".FirebaseApplication"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".AuthUIActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".EditProfileActivity" />
        <activity
            android:name=".MenuTelas.TelaPolicia"
            android:label="Policia" />
        <activity
            android:name=".MenuTelas.TelaBombeiro"
            android:label="Bombeiro"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
            android:name=".MenuTelas.TelaAmbulancia"
            android:label="Ambulancia"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
            android:name=".MenuTelas.TelaMecanico"
            android:label="Mecanico"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
            android:name=".MenuTelas.TelaHotel"
            android:label="Hotel"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
            android:name=".ProfileActivity"
            android:theme="@style/AppTheme.NoActionBar" />

        <service
            android:name=".MyFirebaseMessagingService"
            tools:ignore="ExportedService">
            <intent-filter>
                <action android:name="com.google.firebase.MESSAGING_EVENT" />
            </intent-filter>
        </service>
        <service
            android:name=".MyFirebaseInstanceIDService"
            tools:ignore="ExportedService">
            <intent-filter>
                <action android:name="com.google.firebase.INSTANCE_
            </intent-filter>
        </service>

        <activity
            android:name=".FcmPayloadActivity"
            android:excludeFromRecents="true"
            android:noHistory="true"
            android:theme="@style/Theme.AppCompat.Light.Dialog" />
        <activity
            android:name=".CopyToClipboardActivity"
            android:theme="@android:style/Theme.NoDisplay" />
        <activity
            android:name=".MainActivity"
            android:label="MainActivity"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity android:name=".Local" />
        <activity
            android:name=".Chat"
            android:label="chat"
            android:theme="@style/AppTheme.NoActionBar"></activity>
    </application>

</manifest>
```

Fonte: Elaboração Própria

21. Leitura e gravação dos dados na base de Dados

Para ler e gravar dados na base de dados tem que ter uma instância que é chamada de *DatabaseReference.*

21.1. DatabaseReference

Na figura abaixo mostra como é montada o *DatabaseReference* no Projeto.

Figura 28: Primeiros passos para introduzir DatabaseReference

```
private DatabaseReference databaseReference;

public FirebaseDatabaseHelper(){
    databaseReference = FirebaseDatabase.getInstance().getReference();
}
```

Fonte: Elaboração Própria

22. Leitura e gravação de lista

Método *push ()* é para anexar dados a uma lista em *Apps* de vários utilizadores. Esse Método gera uma chave exclusiva sempre que um novo filho é adicionado a uma referência específico do *Firebase*. Ao usar essas chaves criado automaticamente para cada novo elemento da lista, vários elementos podem adicionar filho no mesmo local simultaneamente sem criar conflitos de gravação. A chave exclusiva gerada por *push ()* é baseada em um caminho de data/hora. Os itens da lista são organizados automaticamente em ordem cronológica.

Usar referência aos dados retornados pelo método push () para receber o valor da chave filho que foi criado automaticamente ou para definir dados para filho. Chamar getkey () em uma referência push () retorna o valor da chave gerada automaticamente.

Ao trabalhar com lista, o App deve detectar eventos filhos em vez de eventos de valor usados para objetos individuais.

Eventos filho são acionados em resposta a operações específicas que ocorrem nos filhos de um node (nó) de uma operação como adição de um novo filho por meio do método *push ()* ou a autorização de um filho pelo método *updatechildren ()*. Cada um desses métodos pode ser útil para detetar alterações em um node específico de um Base de Dados:

- **onChildded() -** Recuperar listas de itens ou detetar adições em uma lista de itens. Este retorno de chamada é acionada filho é uma vez para cada filho existente e sempre que um novo filho é adicionado ao caminho específico. O *DataSnapshot* passado ao listener conte os dados do novo filho;
- **onChilChange() -** Detetar mudanças em itens de uma lista. Este evento é acionado sempre um node filho é modificado, incluindo quaisquer modificações aos descendentes do node filho. O *DataSnapshat* passado ao listener de eventos conte os dados atualizados do filho;
- **onChildRemoved() -** Detetar itens sendo removido de uma lista. O *DataSnapshot* passado ao retorno de chamada do evento conte os dados atualizados do filho;
- **onChildMoved()** - Detetar alterações na classificação dos itens em uma lista ordenada. O evento é acionado sempre que o retorno de chamada *onChilChanget ()* é ativado por uma atualização que causa alterações na ordem do filho. Ele é usado com os dados ordenados com *orderByChild* ou *orderByValue*.

Figura 29: Leitura e Gravação de lista

```
public void createUserInFirebaseDatabase(String userId, FirebaseUserEntity firebaseUserEntity){
    Map<String, FirebaseUserEntity> result = new HashMap<String, FirebaseUserEntity>();
    result.put(userId, firebaseUserEntity);
    databaseReference.child("Utilizador").setValue(result);
}

public void isUserKeyExist(final String uid, final Context context, final RecyclerView recyclerView){
    databaseReference.child("Utilizador").addChildEventListener(new ChildEventListener() {

        @Override
        public void onChildAdded(DataSnapshot dataSnapshot, String s) {
            System.out.println("User login 1 " + dataSnapshot.getKey() + " " + dataSnapshot.getValue());
            List<UserProfile> userData = adapterSourceData(dataSnapshot, uid);
            System.out.println("User login Size " + userData.size());
            RecyclerViewAdapter recyclerViewAdapter = new RecyclerViewAdapter((Activity)context, userData);
            recyclerView.setAdapter(recyclerViewAdapter);
        }

        @Override
        public void onChildChanged(DataSnapshot dataSnapshot, String s) {
            List<UserProfile> userData = adapterSourceData(dataSnapshot, uid);
            System.out.println("User login Size " + userData.size());
            RecyclerViewAdapter recyclerViewAdapter = new RecyclerViewAdapter((Activity)context, userData);
            recyclerView.setAdapter(recyclerViewAdapter);
        }

        @Override
        public void onChildRemoved(DataSnapshot dataSnapshot) {

        }


        @Override
        public void onChildMoved(DataSnapshot dataSnapshot, String s) {

        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });
}

private List<UserProfile> adapterSourceData(DataSnapshot dataSnapshot, String uId){
    List<UserProfile> allUserData = new ArrayList<UserProfile>();
    if(dataSnapshot.getKey().equals(uId)){
        FirebaseUserEntity userInformation = dataSnapshot.getValue(FirebaseUserEntity.class);
        allUserData.add(new UserProfile( header: Helper.NOME,  profileContent: userInformation.getNome()));
        allUserData.add(new UserProfile( header: Helper.EMAIL,  profileContent: userInformation.getEmail()));
        allUserData.add(new UserProfile( header: Helper.IDADE,  profileContent: userInformation.getIdade()));
        allUserData.add(new UserProfile( header: Helper.TELEFONE,  profileContent: userInformation.getTelefone()));
        allUserData.add(new UserProfile( header: Helper.SEXO,  profileContent: userInformation.getSexo()));
    }
    return allUserData;
}
```

Fonte: Elaboração Própria

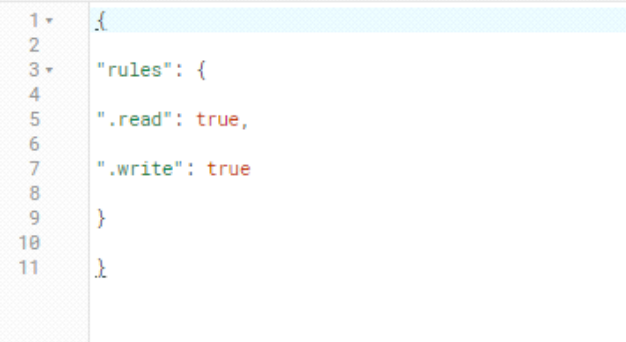
23. Regras de Segurança na Base de Dados do Firebase

23.1. Regras de Segurança Inicial

Para simplificar o processo de leitura e gravação de dados no *Firebase*, as regras de segurança iniciais da Base de Dados são para permitir acesso público.

Na figura abaixo mostra como se aparece as regras depois da modificação, inicialmente se encontra *"false"* depois mudamos para *"true"*.

Figura 30: Regras de Segurança



Fonte: Elaboração Própria

24. Regras da Firebase Realtime Database

As regras do *Firebase Realtime Database* determinam quem tem acesso de leitura e gravação a Base de Dados, como os dados são estruturados e quais índices são definidos.

Essas regras residem nos servidores de *Firebase* e são sempre aplicados automaticamente. Cada solicitação de leitura e gravação só será concluído se as regras permitirem. Por padrão as regras são definidas para acesso completo de leitura e gravação a Base de Dados, somente os utilizadores autenticados. A finalidade é proteger a Base de Dados contra uso indevido até que a pessoa personalize as regras ou configurar a autenticação.

Tipos de Regras:

- **.read** - Descreve se e quando os dados podem ser lidos pelos utilizadores;
- **.write** - Descreve se e quando os dados podem ser gravados;
- **.validate** - define a formatação correta do valor, o tipo de dados e se o valor tem atributo;
- **indexOn** - Especifica um filho como índice para que a ordenação e consulta sejam possíveis.

A primeira etapa na segurança do *App* é a identificação dos utilizadores. Usando o Firebase Authentication para login dos utilizadores no *App*.

Tem suporte ao *Drop-in* em métodos comuns de autenticação como Google, *Facebook*, além do *login* com email e palavra-passe, *login* anônimo, entre outros.

Quando um utilizador autenticar a variável *auth* nas suas regras do *Firebase Database* será preenchida com informações do utilizador. Essa informação inclui um identificador exclusivo (*uid*) e os dados da conta vinculada.

A variável auth pré definida nas regras é nula antes da autenticação. Quando um utilizador é autenticado com *Firebase Authentication*, recebe os seguintes atributos:

- **Provider** - o método de autenticação utilizado ("senha", "anónimo", facebook", "Google", etc.);
- **uid** - um código de utilizador único em todos os provedores;
- **token** - o conteúdo do token de código do *Firebase Auth.*

Validação dos dados

Firebase Realtime não usa esquemas, o que facilita fazer alterações durante o desenvolvimento. É importante manter a consistência dos dados. A linguagem de regras inclui uma regra. Validade para aplicar a lógica de validação com as mesmas expressões usadas pelas regras *.read* e *.write* a única diferença é que como as regras de avaliação não são aplicadas em cascata, todas as regras de avaliação relevantes são avaliadas como Verdadeiro para que a gravação seja permitida.

25. LocationManager

Esta classe fornece acesso aos serviços de localização do sistema. Esse serviço que as aplicações obtenham atualizações periódicas da localização geográfica do dispositivo ou acionem uma aplicação específica intent, quando o dispositivo entra na proximidade de uma determinada localização geográfica.

Instâncias dessa classe deve ser obtida usando *Context.getSystemService (Class)* o argumento *Context.Location_Service.*

Requer *FEATURE_Location*, recursos que pode ser detectado usando *PackageManager.hasSystemFeature (String).*

26. GPS_Provider

Este provedor determina a localização usando satélites. Dependendo das condições, esse provedor pode demorar um pouco para retornar uma correção de local. Requer a permissão *ACESS_FINE_LOCATION.*

O pacote de extras para o provedor de localização GPS pode conter os seguintes pares de chaves/valores:

- Satélites- o número de satélites para derivar ocorrência, exemplo *constant: ”gps”.*

27. Network_Provider

Este provedor determina a localização com base nos dispositivos de torres de telemóveis e pontos de acesso Wifi.

Os resultados são recuperados por meio de uma pesquisa de rede, exemplo *constant:"network".*

28. Passive_Provider

Este provedor pode ser usado para receber possivelmente atualizações de localização quando outras aplicações ou serviços os solicitam sem realmente solicitar os locais. Este provedor retorna locais gerados por outros provedores. Consultar o ***getProvider ()*** método para determinar a origem da atualização do local.

Requer a permissão *ACCESS_FINE_LOCATION*, embora, se o GPS não estiver habilitado, esse provedor se retorne correções grosseiras, exemplo *constant:"passive".*

Abaixo mostra como as funcionalidades são aplicadas no protótipo em si.

Figura 31: Permissão para obter Localização

```
@Override
public void onClick(View view) {
    locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
    if (!locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
        buildAlertMessageNoGps();

    } else if (locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
        getLocation();
    }
}

private void getLocation() {
    if (ActivityCompat.checkSelfPermission( context: TelaPolicia.this, android.Manifest.permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission
            ( context: TelaPolicia.this, android.Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {

        ActivityCompat.requestPermissions( activity: TelaPolicia.this, new String[]{android.Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_LOCATION);

    } else {
        Location location = locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);

        Location location1 = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);

        Location location2 = locationManager.getLastKnownLocation(LocationManager. PASSIVE_PROVIDER);
```

Fonte: Elaboração Própria

29. Sincronizar vertente web com o Firebase

Para adicionar o *Firebase* na parte Web é necessário um novo projeto, mas nesse caso não será necessário, pode ser usado o projeto que o Android já está a utilizar, apenas é utilizado o Firebase SDK é um pequeno *snippet* de código de inicialização.

O *snippet* contém as informações para configurar o SDK para *JavaScript* do *Firebase* para o uso do *Authentication*, do *Cloud Storage*, do *Realtime Database* e do *Cloud Firestore*. Reduz a quantidade de código usado pela aplicação.

Figura 32: Código de inicialização para sincronizar Firebase com a parte Web

```
<script src="https://www.gstatic.com/firebasejs/4.10.1/firebase.js"></script>
<script>
  // Initialize Firebase
  // TODO: Replace with your project's customized code snippet
  var config = {
    apiKey: "<API_KEY>",
    authDomain: "<PROJECT_ID>.firebaseapp.com",
    databaseURL: "https://<DATABASE_NAME>.firebaseio.com",
    storageBucket: "<BUCKET>.appspot.com",
    messagingSenderId: "<SENDER_ID>",
  };
  firebase.initializeApp(config);
</script>
```

Fonte: https://firebase.google.com/docs/web/setup#top_of_page2

Componentes que podem ser instalados individualmente:

- **firebase_app:** o cliente *Firebase* principal (obrigatório);
- **firebase_auth:** *Firebase Authentication* (opcional);
- **firebase_database:** *Firebase Realtime Data base* (opcional);
- **firebase_firestore:** *Cloud Firestore* (opcional);
- **firebase_storage:** *Cloud Storage* (opcional);
- **firebase_messaging:** *Firebase Cloud Messaging*.

30. Permissões e Verificações de conectividade

Para criar uma aplicação com acesso a ligação a rede é importante incluir no manifesto, *AndroidManifest.xml* do projeto Android as permissões que permitem que as aplicações abram *sockets* de rede e que possam aceder a informações sobre outras redes.

Abaixo se segue um exemplo da permissão da internet, essa permissão se encontra no AndroidManifest.xml.

Figura 33: Permissão e Verificação de Conectividade

```
<uses-permission android:name="android.permission.INTERNET" />
```

Fonte: Própria

Para verificação da rede, usa-se normalmente as seguintes classes:

- **ConnectivityManager** - resolve *queries* sobre o estado da conectividade de rede. Também notifica as aplicações quando existem mudanças da conectividade de rede;
- **NetworkInfo** - desenvolve o estado de uma interface de rede de um determinado tipo.

31. Solicitar permissões dos utilizadores

Para receber atualizações de um local de Network_Provider ou GPS_Provider, é necessário solicitar permissões do utilizador declarando a *ACCESS_COARSE_LOCATION* ou a *ACCESS_FINE_LOCATION*, respectivamente no arquivo manifesto do Android. Sem essas permissões a aplicação falha no tempo de execução solicitar atualização do local.

Ao usar *Network_Provider* e *GPS_Provider*, precisa solicitar apenas a *ACCESS_FINE_LOCATION*, inclui para ambos os provedores. Permissão para *ACCESS_COARSE_LOCATION* permite acesso apenas a *Network_Provider*.

Tem que ter uma atenção para aplicações que tem como alvo Android 5.0 (nível API 21) ou superior, tem que declarar a aplicação como é usado, *android.hardware.location.network* ou de outra forma *android.hardware.location.gps*, recurso de *Hardware* no arquivo de manifesto, dependendo se a aplicação recebe atualizações de localização de *Network_Provider* ou a partir de *GPS_Provider*. Se a aplicação receber informações do local de qualquer uma dessas origens de provedores de localização, precisa deslocar que a aplicação usa recursos de Hardware no manifesto da aplicação.

Figura 34: Solicitar Permissão dos Utilizador

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Fonte: Elaboração Própria.

CAPÍTULO V

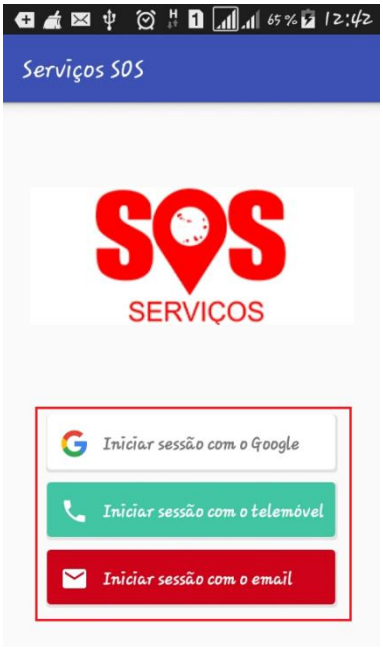
32. Funcionamento do Protótipo

O funcionamento do protótipo é apresentado de acordo com as funcionalidades do sistema desenvolvido em si, para um melhor entendimento no caso de utilização.

32.1. Parte da Aplicação

Ao executar a aplicação a primeira parte que aparece é o *login*, este *login* pode ser feito de três formas diferente. Tem possibilidades de escolher como fazer esse *login*.

Figura 35: Opções de Login



Fonte: Elaboração Própria

Primeira opção é se o utilizador pretende fazer o *login* com uma conta Google (*gmail*), faz uma busca automática para poder sincronizar a sua conta com a aplicação.

Figura 36: Opção para iniciar sessão com o Google

Fonte: Elaboração Própria

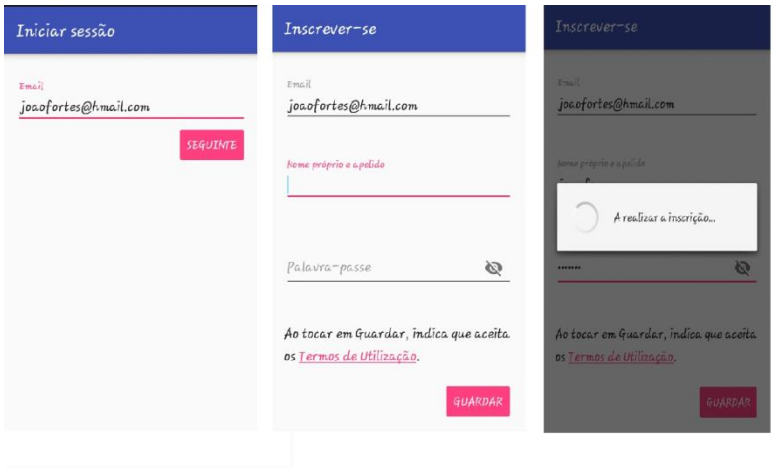
Segunda Opção é introduzir o número do Telemóvel onde através desse número é gerado um código automaticamente, depois da validação deste código o utilizador já tem permissão para começar a utilizar a aplicação.

Figura 37: Opção para iniciar sessão com o número do Telemóvel

Fonte: Elaboração Própria.

A terceira e última opção é colocar um Email, ao introduzir o email é pedido o seu nome próprio de seguida uma palavra passe para validar esse processo de *login*.

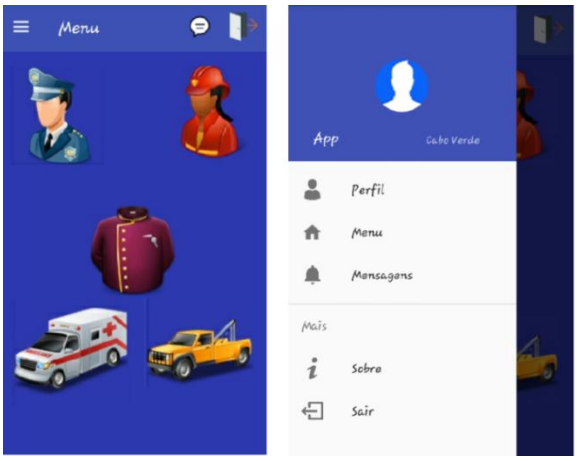
Figura 38: Opção para iniciar sessão com outro tipo de Email



Fonte: Elaboração Própria

A seguir o menu principal é apresentado os serviços, dentro desse menu principal há um submenu onde temos algumas opções tais como: Perfil, Menu, Mensagem, Sobre e Sair.

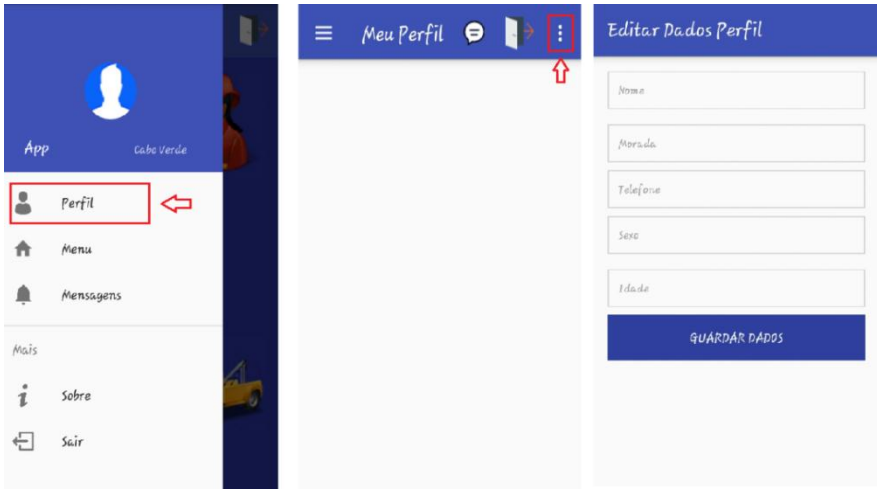
Figura 39: Menu Principal e o submenu da Aplicação



Fonte: Elaboração Própria

Antes de começar a utilizar a aplicação é preciso introduzir alguns dados do seu perfil, vai encontrar uma opção na parte superior onde pode aceder um pequeno formulário.

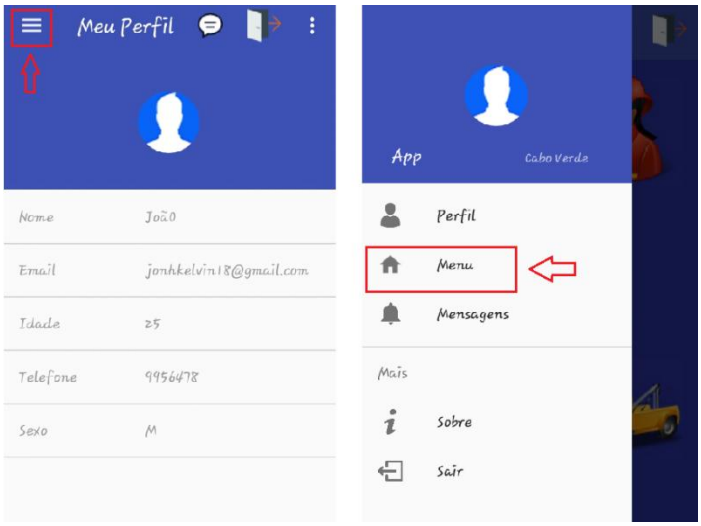
Figura 40: Introdução de Dados do Perfil



Fonte: Elaboração Própria

Com os dados preenchidos regressa ao menu principal através do submenu.

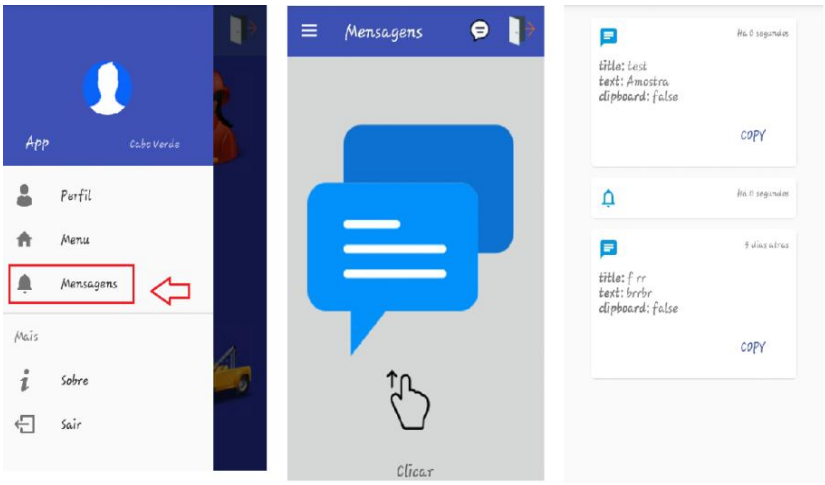
Figura 41: Regresso ao Menu Principal



Fonte: Elaboração Própria

No submenu se encontra uma opção “Mensagens “ onde pode consultar se tiver recebido algum SMS no dispositivo.

Figura 42: Verificação de SMS



Fonte: Elaboração Própria

De seguida a escolha do Serviço que pretende pedir socorro.

Ao escolher um serviço, vai encontrar algumas opções, antes de escolher a (s) opção (s) é necessário clicar no botão “ Obter Localização”, só depois clicar na opção pretendida de seguida as informações da posição serão enviadas juntamente com o pedido.

Um dos serviços escolhidos para exemplificar foi Serviço de Polícia.

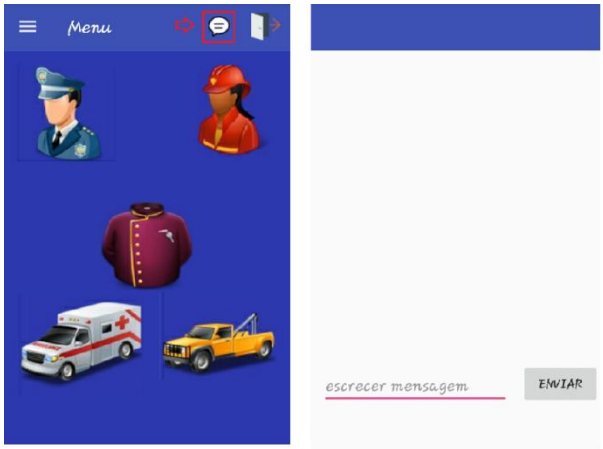
OBS: Esse procedimento é feito para que a entidade possa obter a posição atual no Mapa juntamente com o pedido de socorro.

Figura 43: Envio das Ocorrências

Fonte: Elaboração Própria

O utilizador pode trocar conversas com a entidade responsável através de um pequeno chat que se encontra alojado na parte superior do menu principal.

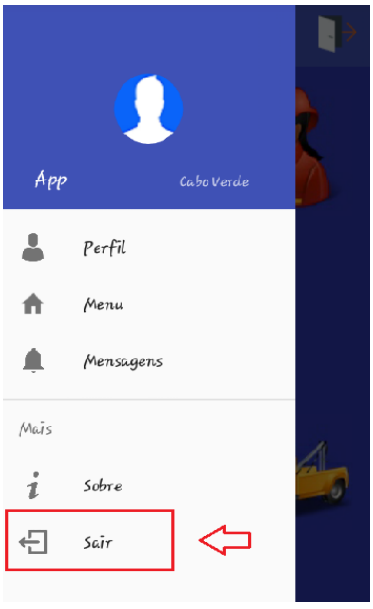
Figura 44: Chat da Aplicação



Fonte: Elaboração Própria

No submenu tem uma opção "Sair" onde pode sair da aplicação sem fazer *logout*.

Figura 45: Sair da Aplicação Temporariamente



Fonte: Elaboração Própria.

Por fim tem uma opção onde podemos fazer logout onde saímos da por completo.

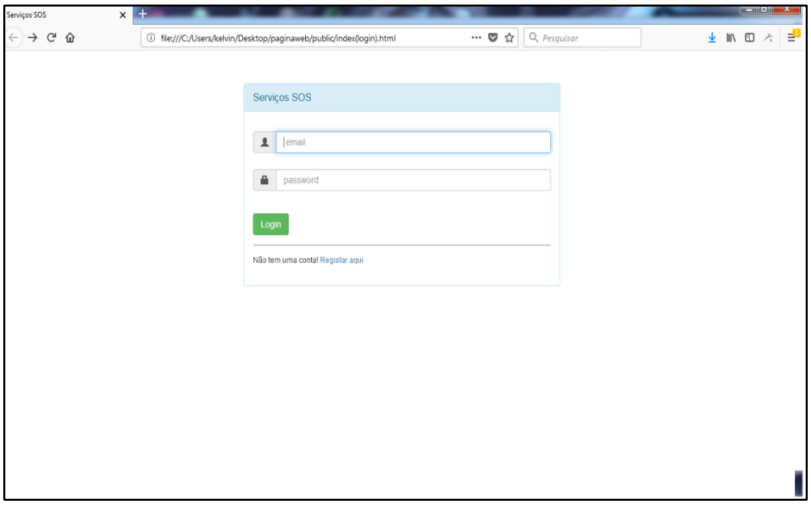
Figura 46: Opção Logout

Fonte: Elaboração Própria

32.2. Parte Web

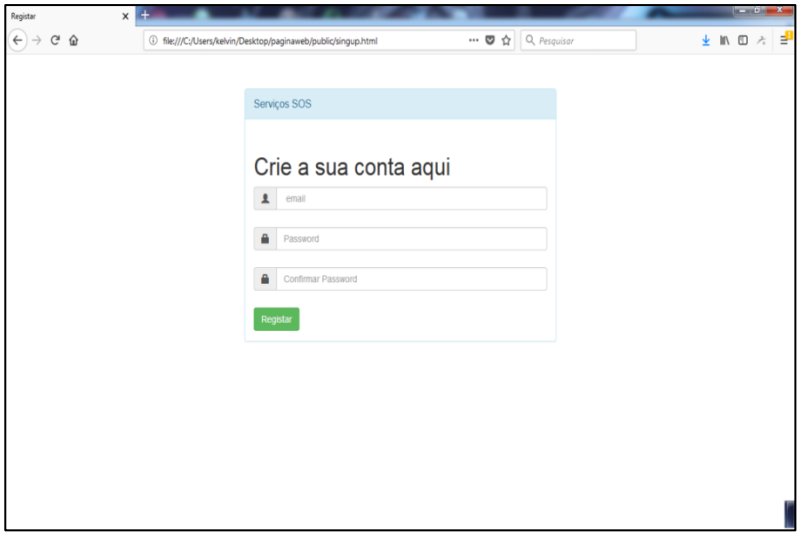
Apresentação de outra parte do protótipo que será uma página Web onde primeiramente a entidade faz *login*, caso estiver autenticado, se não terá que criar uma conta.

Figura 47: Login na Página Web



Fonte: Elaboração Própria

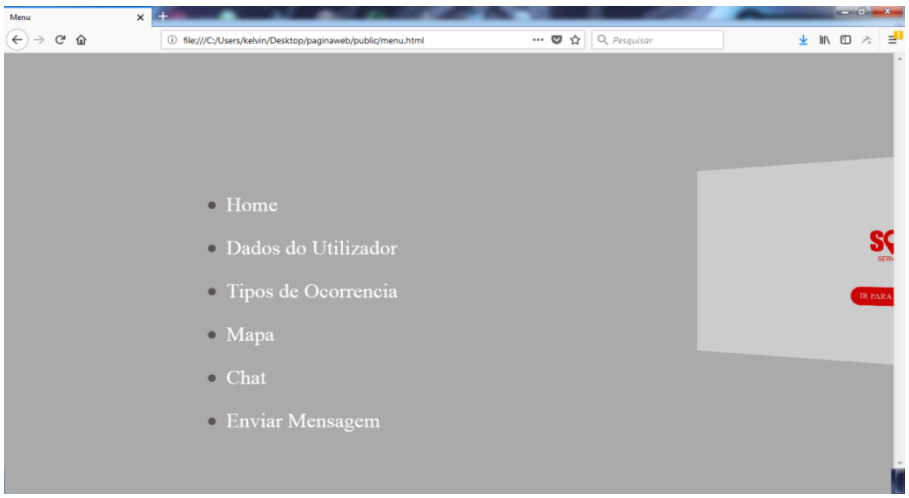
Figura 48: Criação de uma conta



Fonte: Elaboração Própria

Depois de fazer o login é apresentado um menu, onde encontra algumas opções para dar seguimento aos pedidos solicitados.

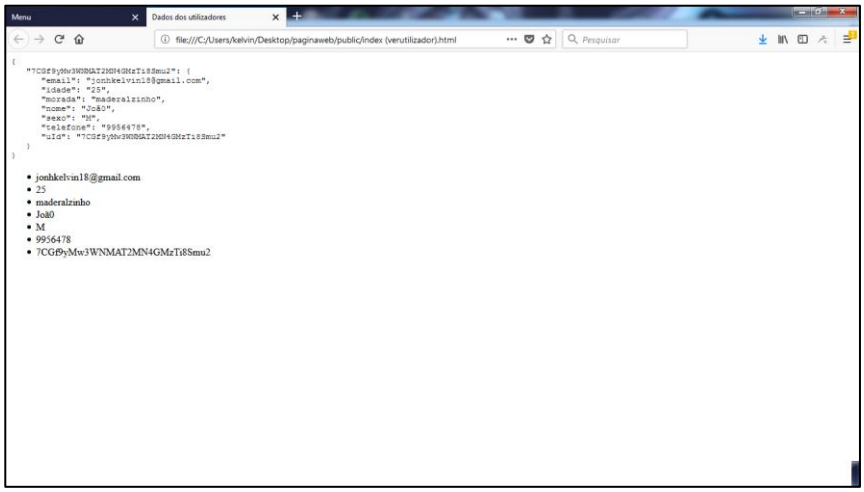
Figura 49: Menu da Página Web



Fonte: Elaboração Própria

Segue-se a opção “ Dados do utilizador” nesta parte a entidade vai ter acesso a dados do perfil do utilizador.

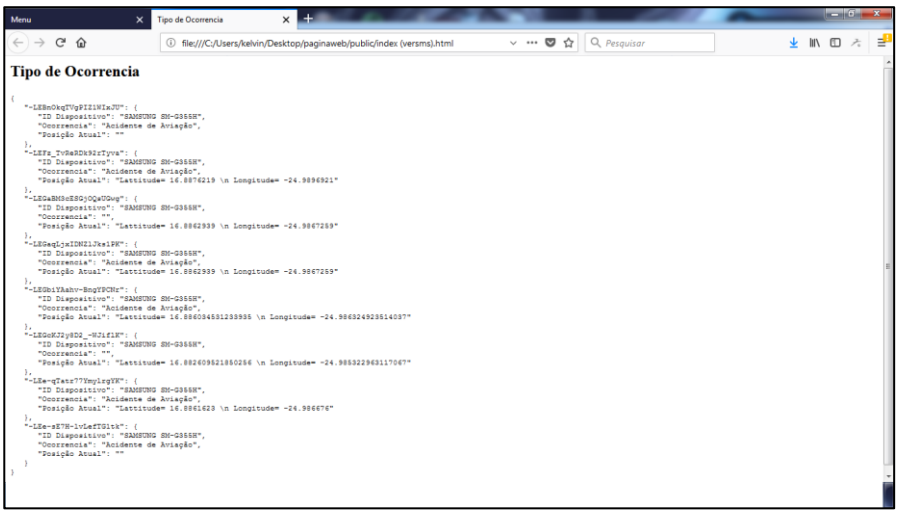
Figura 50: Dados do Perfil do Utilizador



Fonte: Elaboração Própria

A opção “tipo de ocorrência ” se encontra o (s) tipo (s) de ocorrência (s) que é solicitado pelo utilizador, referência o Dispositivo caso tenha outro e a posição que se encontra.

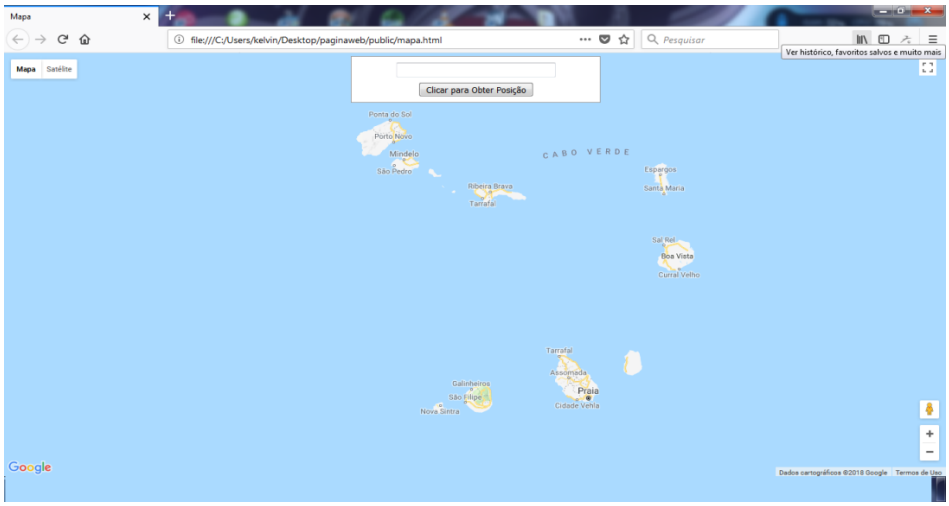
Figura 51: Dados das Ocorrências



Fonte: Elaboração Própria

De seguida a Opção “Mapa” onde tem um pequeno quadro onde é introduzido as referências de localização que é disponibilizada pelo utilizador.

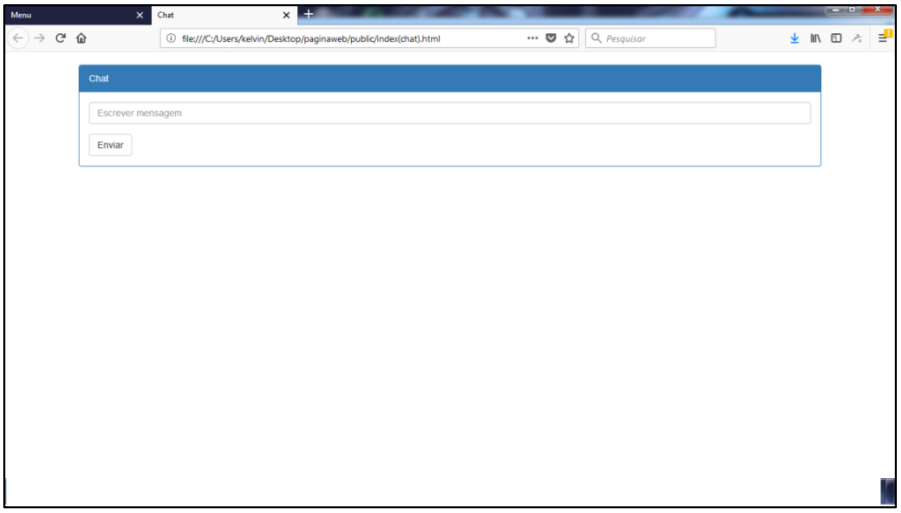
Figura 52: Mapa para obter a posição atual do utilizador



Fonte: Elaboração Própria

Apresentação de um pequeno Chat onde a entidade pode se comunicar com o utilizador através de uma conversa em tempo real.

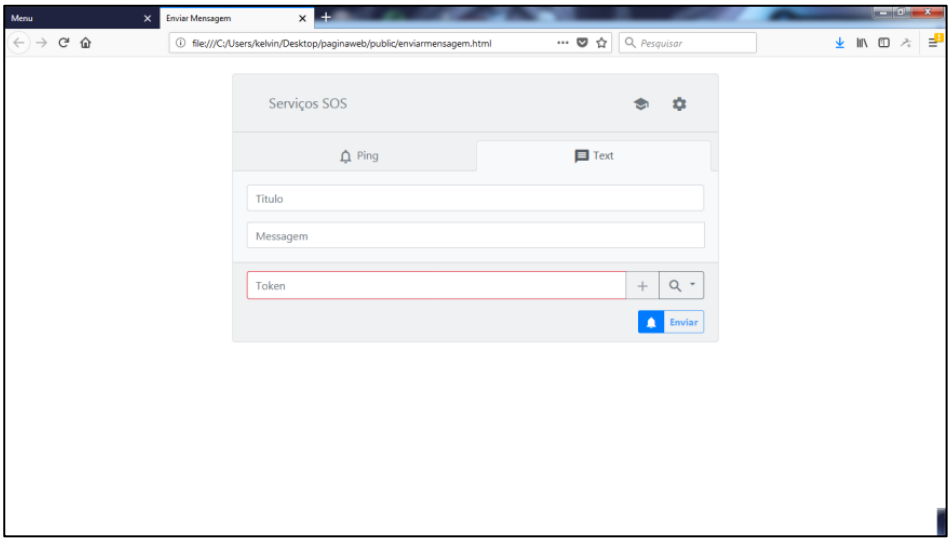
Figura 53:Chat da página Web



Fonte: Elaboração Própria

Nesta parte a entidade pode enviar SMS para aplicação do utilizador caso haja necessidade.

Figura 54: Envio de SMS para Aplicação



Fonte: Elaboração Própria

CAPÍTULO VI

33. Conclusão

Ao longo do desenvolvimento deste trabalho percebe-se que o uso de novas tecnologias para a criação da aplicação foi um desafio bastante interessante e vantajoso.

Este projeto foi criado para dar resposta às necessidades dos cidadãos e de uma certa forma também para facilitar as entidades competentes.

Como desenvolvimento desta aplicação o utilizador e a entidade podem ficar mais interligados entre si, de acordo com as funcionalidades que a aplicação traz, facilita a comunicação entre ambas as partes.

Na aplicação o utilizador encontra opções que lhe pode ajudar em caso de emergência. Dentro dessa vertente conta um conjunto de dados que é disponibilizado pelo próprio utilizador, com esses dados disponibilizados a entidade pode dar seguimento as ocorrências, sabendo que é fornecido a posição atual do utilizador, essa tecnologia facilita a entidade na hora de se deslocar e também decidir quais os procedimentos que deve tomar, essas informações são vistas em tempo real.

Como se trata de desenvolvimento de um protótipo é difícil dizer já está "finalizado". Sempre surgem novas tecnologias que podem ser introduzidas ou ainda ideias de funcionalidades.

Capítulo VII

34. Trabalhos Futuros

Com desenrolar do projeto surgiram ideia que futuramente podem ser aplicadas no desenvolvimento do Protótipo, tais como:

Na parte da Aplicação

- Estender a aplicação para outras plataformas mobile a *iOS* e a *Windows Fome*;
- Criar uma opção onde o utilizador possa fazer Videochamadas para a Entidade caso for necessário;
- Colocar um sistema de notificação, quando o utilizador recebe mensagem pelo Chat da aplicação;
- Permitir localizar no mapa as Entidades que se encontram mais perto do utilizador.

Na página Web

- Receber notificação tanto na Chat, também na parte onde é disponibilizado as informações do utilizador sobre as ocorrências que é disponibilizada para a Entidade responsável pelo socorro;
- Permissão para fazer videochamada para a aplicação;
- Tratamento dos dados recebidos (estética), ou seja uma apresentação mais apropriada.

Referência Bibliográfica

Silva, Luciana, Android Programando Passo-Passo, 7º Edição.

Caelum, (ensino e inovação) Desenvolvimento Web com HTML, CSS e JavaScript.

HandBook de TI para concursos, O Guia Definitivo.

Silva, Alberto e Vieira, Carlos (2011). UML, Metodologia e Ferramentas CASE, 1º Edição.

Griffiths, Dawn e Griffiths, David, Android Development.

Nunes, Mauro e O'Nell, Henriques (2004). Fundamental de UML, 4º Edição.

Lecheta, Ricardo R (2016) Google Android, Aprenda a criar aplicativos para dispositivos móveis com SDK, 5 ª Edição.

Deitel, Paul e Harvey (2010) Java como Programar 8ª Edição.

Deitel, Paul e Harvey (2016), Android 6 para Programadores - 3ª Edição: Uma Abordagem Baseada em Aplicativos.

Carvalho, Kleber Rodrigo (2012), Aplicativos Web Pro Android.

Smyth, Neil (2015), Android Studio Development Essentials- Android 6 Edition ©.

Wiley, John Sons, Inc. (2015), Android™ Application Development All-in-One For Dummies®, 2nd Edition.

Figueiredo, Bruno (2004) Web Design: Estrutura, Conceição, e Produção de Sites Web. Lisboa: FCA.

Componentes da aplicação Android. Disponível em: <www.w3big.com/pt/android/android-application-components.html> [Consultado em 10-01-2018, 16:10];

Ciclo de Vida de uma Aplicação. Disponível em: <http://celeiroandroid.blogspot.com/2011/02/o-que-e-o-android.html> [Consultado em 25-01-2018, 12:25];

Versões da plataforma Android. Disponível em: <www.techtudo.com.br/noticias/2017/11/dez-anos-do-android-relembre-as-versoes-do-sistema-do-google-para-celular.html> [Consultado em 25-01-2018, 12:40];

Coordenadas Geográficas. Disponível em: <www.sogeografia.com.br/Conteudos/GeografiaFisica/coodenadas.geo>[Consultado em 25-04-2018, 11:25];

Linguagem Java. Disponível em: <www.info.wester.com/lingjava.php> [Consultado em 25-01-2018, 12:25];

Note.js. Disponível em: <https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm> [Consultado em 20-04-2018, 15:40];

Linguagem HTML Disponível em: <www.infoescola.com> [Consultado em 25-01-2018, 14:15];

Linguagem CSS. Disponível em: http://portalwebdesigner.com/programacao/css/> [Consultado em 25-01-2018, 10:00];

JSON.Disponível em: <https://developer.mozilla.org/enUS/docs/learn/javascript/object/json> [Consultado em 10-02-2018, 10:55];

Androide Studio. Disponível em: < www.androidpro.com.br/android-sdk > [Consultado em 15-02-2018, 21:30];

Visual Paradigm. Disponível em: < www.sotfware.com.br/p/visual-paradigm#product-description> [Consultado em 23-03-2018, 9:00];

Notepad++.Disponível em: < https://notrpad-plus-plus/features> [Consultado em 28-03-2018, 17:00];

Firebase. Disponível em: **<** www.mecreiros.com/firebase-o-que-e-e-como-funciona> [Consultado em 09-04-2018, 14:45];

Firebase Realtime Database. Disponível em: <https://firebase.google.com/docs/database/> [Consultado em 10-04-2018, 17:30];

Firebase Authentication. Disponível em: < https://firebase.google.com/docs/auth/> [Consultado em 11-03-2018, 18:00];

Firebase Cloud Massaging. Disponível em: < https://firebase.google.com/docs/cloud-messaging> [Consultado em 30-04-2018, 19:30];

Firebase Hosting. Disponível em: < www.firebase.google.com/docs/hosting>[Consultado em 02-05-2018, 16:30];

Notification. Disponível em: < www.treinaweb.com.br/blog/firebase-descubra-no-que-esta-plataforma-pode -te-ajudar/> [Consultado em 25-04-2018, 12:00];

Protótipo. Disponível em: <http://mds.cultura.gov.br/core.base_rup/guidances/concepts/prototypes_9D1E67A.html> [Consultado em 18-05-2018, 11:00];

< https://neigrando.wordpress.com/2013/06/04/usando-prototipos-para-dar-forma-as-ideias/> [Consultado em 10-06-2018, 20:00];

Arquivo AndroidManifest.xml no Android. Disponível em: <www.javatpiont.com/AndroidManifest-xml-file-in-android>[Consultado em 10-06-2018, 20:10];

AngularJS. Disponível em: < https://www.portalgsti.com.br/angularjs/sobre/ > [Consultado em 08-06-2018, 16:00].

Anexos

TERMO DE RESPONSABILIDADE DE ORIENTAÇÃO

UNIVERSIDADE DO MINDELO

Sapientia Omnium Potentior Est

TERMO DE RESPONSABILIDADE de ORIENTAÇÃO

Eu, Samuel Santos de Lima, grau: Licenciado.
declaro que o aluno Kilvin Jonhson Nascimento da Cruz, N.º 2848
Finalista do curso de Informática de Gestão, realizou sob a minha orientação o Trabalho de Conclusão Curso/Monografia/Relatório de Estágio/Projeto de Licenciatura intitulada: "Serviços SOS: Aplicação Móvel para Pedido de Socorro".
e que a mesma foi desenvolvida de acordo com as Normas de Elaboração e Apresentação dos TCC's da **UNIVERSIDADE DO MINDELO** e reúne todas as condições para a sua apresentação e defesa.

Mindelo, 22 de Junho de 2018

O Orientador

Rua Patrice Lumumba, CP 648 – 2110 – Mindelo – São Vicente – CABO VERDE
https://uni-mindelo.edu.cv – e-mail geral@uni-mindelo.edu.cv – Telefone: +238.2326810 – Fax: +238.2325132
NIF: 562770755

TERMO DE ACEITAÇÃO

UNIVERSIDADE DO MINDELO

Sapientia Ars Vivendi

TERMO DE ACEITAÇÃO DO TEMA PARA O DESENVOLVIMENTO DO TCC – TRABALHO DE CONCLUSÃO DE CURSO

Eu, Kelvin Jonhson Nascimento da cruz, Aluno N.º 2848 do 4º Ano do Curso de Licenciatura em Informática de Gestão da UNIVERSIDADE DO MINDELO, declaro que aceito desenvolver o meu TCC para conclusão do curso, com o Tema: Serviços SOS: Aplicação Móvel para Pedido de Socorro,

de acordo com os Regulamentos e com as Normas vigentes na UNIVERSIDADE DO MINDELO, comprometendo a entregar o referido trabalho em 3 (três) exemplares e um CD/DVD, no prazo fixado pelo Conselho Científico do DEPARTAMENTO DE ENGENHARIA E RECURSOS DO MAR.

Proponho ainda que seja designado como meu Orientador o Sr.: Samuel Santos de Alva, Licenciado/Mestre/Doutor em Informática de Gestão.

Mindelo, 22 de Junho de 2018

Kelvin da cruz

(O Estudante)

Aceitação da Orientação

(O Orientador)

Sujeito a validação pela coordenação, anexando o CV do Orientador (só e (uma) 1).

Rua Patrice Lumumba, CP 648 – 2110 Mindelo – São Vicente – CABO VERDE
https://uni-mindelo.edu.cv – e-mail geral@uni-mindelo.edu.cv – Telefone: +238.2326810 – Fax: +238.2325132
NIF: 562770755

DECLARAÇÃO DE AUTORIZAÇÃO DE ARQUIVO E DIVULGAÇÃO

UNIVERSIDADE DO MINDELO

DECLARAÇÃO DE AUTORIZAÇÃO DE ARQUIVO E DIVULGAÇÃO

NOME
Kelvin Jonhson Nascimento da cruz

BI/PASSAPORTE: 358298 | TELEMÓVEL: 9822707 | E-MAIL: JonhKelvin18@gmail.com

Nº DE ESTUDANTE DA UM
2848

RELATÓRIO [X] TESE LICENCIATURA [ ] TESE MESTRADO [ ] TESE DOUTORAMENTO [ ] OUTRO [ ]

DATA DE CONCLUSÃO

RAMO/ESPECIALIDADE
Licenciatura em Informática de Gestão

TÍTULO
Serviços SOS: Aplicação Móvel para Pedido de Socorro

ORIENTADOR(ES)
Samuel Santos de Lima

[X] NÃO AUTORIZO
[ ] AUTORIZO

Declaro, para os devidos efeitos, que concedo gratuitamente à Universidade do Mindelo autorização para arquivar e tornar acessível aos interessados, nomeadamente através do seu repositório institucional, o trabalho supra identificado, que disponibilizo no formato abaixo indicado. A subscrição da presente declaração não implica a renúncia à titularidade dos direitos de autor a direito de usar a obra nos trabalhos futuros os quais são pertença do seu criador intelectual.

FORMATO
[ ] Papel [ ] CD/DVD

OBSERVAÇÕES

	DATA/ NOTA	ASSINATURA SAA
Data Entrega SAA	.....	.....
Data Entrega Coordenação	.....	.....
Data Defesa	.....	.....
Nota Defesa	.....	.....
Data Entrega Biblioteca e Arquivo	.....	.....

Rua Patrice Lumumba, CP 648 – 2110 Mindelo – São Vicente – CABO VERDE
https://uni-mindelo.edu.cv – e-mail geral@uni-mindelo.edu.cv – Telefone: +238.2326810